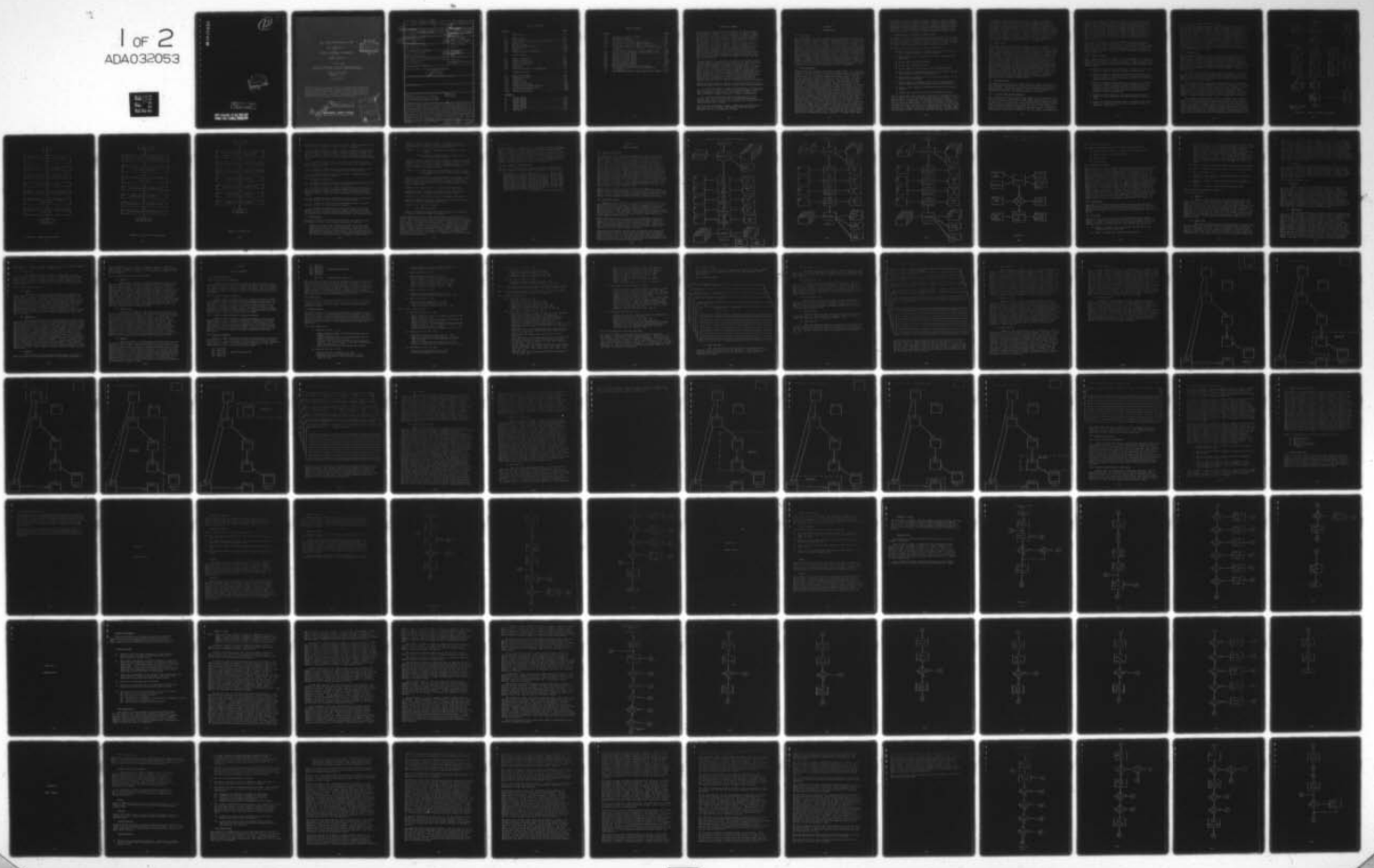
AD-A032 053 MANAGEMENT SCIENCE SYSTEMS INC FALLS CHURCH VA F/G 5/9
THE RELATIONSHIP OF MANPOWER AND OPERATIONAL TASKING FOR SHIPS:--ETC(U)
JUL 76 D R BENBENNICK, P P BRUCE, E W LULL N00014-76-C-0473
UNCLASSIFIED 357602 NL

1 OF 2
ADA032053



FC

THE RELATIONSHIP OF MANPOWER & OPERATIONAL TASKING FOR SHIPS

FINAL REPORT

on the development of the SHIP ROC/Watchstation subsystem of the Navy Manpower Requirements System

David R. Benbennick
Patricia P. Bruce
Edward W. Lull

24 July 1976

This development work was conducted under the Operations Research Program of the Office of Naval Research under Contract N00014-76-C-0473. Funding support and technical guidance were provided by the Navy Manpower and Material Analysis Center, Atlantic.

MANAGEMENT SCIENCE SYSTEMS
7700 Leesburg Pike • Falls Church, Virginia 22043 • (703) 821-6980

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
THE RELATIONSHIP OF MANPOWER AND OPERATIONAL TASKING FOR SHIPS; Final Report on the Development of the SHIP ROC/Watchstation Subsystem of the Navy Manpower Requirements System.

5. TYPE OF REPORT & PERIOD COVERED
FINAL REPORT.
24 Nov 1975 — 24 July 1976

6. PERFORMING ORG. REPORT NUMBER
357602

7. AUTHOR(s)
David R. Benbennick
Patricia P. Bruce
Edward W. Lull

8. CONTRACT OR GRANT NUMBER(s)
N0014-76-C-0473 NEW

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Management Science Systems Inc.
7700 Leesburg Pike
Falls Church, Virginia 22043

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
NR047-127

11. CONTROLLING OFFICE NAME AND ADDRESS
Office of Naval Research
Department of the Navy
Arlington, Virginia 22217

12. REPORT DATE
24 July 1976

13. NUMBER OF PAGES
166

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*
Navy Manpower and Material Analysis Center, Atlantic
U. S. Naval Station
Norfolk, Virginia 23511

15. SECURITY CLASS. *(of this report)*
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*
Approved for public release; distribution unlimited.

12 168 p.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*
Manpower
Navy Manpower Planning System (NAMPS)
Navy Manpower Requirements System (NMRS)
Required Operational Capabilities (ROC)
Watchstation
Watchstander

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*
Tasking levels for ships are expressed as Required Operational Capability (ROC), which are published by the Office of the CNO for each class of ships. The Billet Derivation (BILDER) process of NMRS requires detailed watchstation and watchstander information as an operational workload input to process with maintenance workload requirements and other factors to determine billets. Thus, the need to relate the ROC for ships with manpower as well as provide BILDER with the necessary operational workload information were merged into the requirements for a subsystem of NMRS called the SHIP/ROC Watchstation Module. The system was developed, tested, documented, and delivered to the user, NAVMMACLANT, for use as a subsystem of NMRS.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED 393 643
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

One of the key functional requirements of the Navy Manpower Planning System (NAMPS) is the capability to rapidly determine the manpower impact of varying the levels of tasking on Navy activities. These tasking levels for ships are expressed as Required Operational Capabilities (ROC), which are published by the Office of the CNO for each class of ships. This developmental function falls into the realm of the Navy Manpower Requirements System (NMRS). The Billet Derivation (BILDER) process of NMRS requires detailed watchstation and watchstander information as an operational workload input to process with maintenance workload requirements and other factors to determine billets. Thus, the need to relate the ROC for ships with manpower as well as provide BILDER with the necessary operational workload information were merged into the requirements for a subsystem of NMRS called the SHIP ROC/ Watchstation Module.

To relate the ROC to manpower, each element of the ROC, called a suboperational capability (i.e. Steam at full power; Provide direct gunfire support for amphibious landing; etc.) was examined with regard to each Minor Functional Area (i.e. Signal Bridge; Sonar Control; Damage Control Central; etc.) to determine which areas had to be manned to support the tasking. After completing a matrix of suboperational capabilities versus minor functional areas, the relevant areas were reexamined with watchstation listings to pinpoint which individual watchstations were absolutely necessary to support the tasking. These minimum watchstation requirements were then tied to rate/rating/Navy Enlisted Code by standards previously developed.

Using this methodology to acquire the necessary data to proceed, the ROC/WS Module was designed to perform the following functions:

a. Given the input of a ship identification, call the ship ROC from the data base, translate the suboperational capabilities into watchstation requirements, and output to BILDER via an input processor, minimum operational manpower requirements.

b. Entering the Module with ship identification and a revised ROC, output a revised set of minimum watch-related requirements without changing the permanent data associated with the particular ship or class.

The system was developed, tested, documented, and delivered to the user, the Navy Manpower and Material Analysis Center, Atlantic for use as a subsystem of NMRS.

SECTION 1

INTRODUCTION

1.1 Background

For many years it has been recognized in the Navy that the most expensive resource to provide is also the most difficult one to justify at the detail level - manpower. Quantifying the requirements for the numerous occupational specialties at various skill levels in an environment as dynamic as the Navy is a very challenging undertaking. Responding to the need to present a strong manpower case in a period of shrinking budgets, the Deputy Chief of Naval Operations (Manpower) (OP-01) initiated the development of the Navy Manpower Planning System, NAMPS. Although the system would have to be large and complex, the philosophy would be relatively simple. Operational and functional capabilities would be related to manpower such that required capabilities input to the system would yield manpower requirements as output. Although there are other significant features to NAMPS, such as the Alternatives Generator and the Personnel and Training Projections Models, the most basic and important function of NAMPS is the pricing out of required capabilities in terms of manpower.

1.2 NAMPS Functions

Within NAMPS, the Manpower Determination Model (MDM) is the dynamic element in the determination of manpower requirements and resolution of alternatives. The operational requirements (Required Operational Capabilities - ROC; Projected Operational Environment - POE; Required Functional Capability - RFC) are the independent variables for manpower requirements determination. The MDM uses the operational requirement input to retrieve from the Manpower Reference Model (MRM) the manpower required to support each element of operation. It aggregates the segments of required manpower to produce the overall (projected) manpower requirement. With this being accomplished in an automated system, it permits the identification of these requirements in sufficient qualitative detail that personnel and training plans may be based upon a defensible manpower package. This also results in significant improvement in the quality of manpower cost estimates. The feedback loops provided in the system are also of great importance to the operation. In those cases where constraints are encountered which signify that the manpower plan is not achievable, timely action can be taken to revise the requirements as necessary during the same planning cycle. The primary constraints to impact upon the manpower plan are normally funding and end strength limitations. However, personnel (accession planning, structure and promotion planning, loss planning, etc.) and training (rating group entry training, long lead time training plans, highly specialized training plans, etc.) considerations can also provide the need for feedback to manpower planning. In these cases, where

sufficient personnel management actions cannot be taken to effect a match between the projected personnel inventory and the manpower plan, the system feeds back the relevant information to the MDM. The MDM passes this information to the Alternative Generator, which manipulates the operational requirement inputs to derive feasible sets of plans, considering the constraints. The new inputs are passed through the MDM to produce revised projected requirements.

1.3 Navy Manpower Requirements System (NMRS)

The original concept of the MRM was somewhat flexible - it could be either a data base or a dynamic model (or a combination of both). Development of NMRS has combined the functions of the MDM (dynamic) and MRM (reference), so that the current design does not have NMRS as a part of the reference model, but rather the opposite.

The NMRS design provides the following capabilities:

- Store reference data for computing manpower requirements (e.g., maintenance data, staffing tables).
- Store operational requirements data (e.g., squadron POE, shore RFC).
- Accept user execution commands.
- Accept user variability data.
- Compute manpower requirements (on an activity basis).
- Store manpower requirements (on an activity basis) linked to the user commands and variability.
- Store manpower requirements (on an activity basis) as approved and promulgated.
- Produce draft requirements documents for analyst/fleet review (including working papers back-up documentation).
- Produce final manpower requirements documents for promulgation.
- Answer user queries at detailed and aggregate levels.

These capabilities serve the Navy in two areas - documenting the actual requirements and projecting potential requirements. In documenting actual requirements, the billet derivation capability of NMRS serves as a substitute for existing automated or manual systems of manpower computation. The crucial capability of NMRS that is lacking in other systems is the availability of all the data in a single data base, for purpose of aggregation and queries. Additionally, as a substitute for the existing systems, the NMRS computation process offers refinements and efficiencies, which could be

expected to reduce the cost (dollar and manpower) of operation. In projecting potential requirements, the integration of the billet derivation process into a system including the input and output data gives the user a quick turn-around at a relatively low cost. Parts of NAMPS will not be automated in the near future - and some parts of the process may prove to be infeasible or uneconomical to ever automate. The purpose of this discussion is to place the NMRS in perspective for the near term as well as the far term. NMRS is not simply a data base - a part of the Manpower Reference Model. If sufficient emphasis is placed upon its development, it will be the heart of the automated NAMPS for some time to come; the solid foundation upon which NAMPS can be developed.

1.4 The Problem

Throughout these discussions, one theme continues to prevail: Manpower is a resource needed to buy capabilities. In the case of ships, this refers to the ROC, the Required Operational Capabilities which are published by the Office of the Chief of Naval Operations as guidance regarding the missions the ship should be capable of performing. Even as the ship should not be assigned a surface gunnery mission if it had no guns, it also should not be expected to conduct surface gunnery with no Gunners Mates. However, in the existing documentation systems there is no direct or indirect relationship which ties a ship's ROC to manpower requirements. As a result, there was no existing methodology which could be applied in the development of the Navy Manpower Requirements System (NMRS) which could establish the link between required capabilities and manpower. Management Science Systems proposed to identify the relationships between the ROC and manpower, and develop, test, and deliver a subsystem of NMRS which would automate the function, and be compatible with the associated subsystems and the data base which were being developed at the Navy Manpower and Material Analysis Center, Atlantic (NAVMMACLANT).

1.5 The Environment

In developing the approach for this subsystem, which was called the Ship ROC/Watchstation Module, several existing situations had to be considered. Primarily, the decisions regarding NMRS which had already been made were significant, as were other decisions which emerged subsequent to the initial design of this module.

1.5.1 General NMRS System Concepts

The Navy Manpower Requirements System (NMRS) is an automated system being developed by NAVMMACLANT, under sponsorship of OP-01, to serve as the requirements generator for the Navy Manpower Planning System. In this role, it can be used to produce billet requirements (for the NAMPS Manpower Reference Model - MRM) and also to investigate manpower requirements based on hypothetical inputs, i.e., to simulate manpower requirements for different levels of tasking. The billet requirements, which will be stored in the data base, will also serve

to produce manpower requirements documents for review and promulgation. The "customers" of the system will include OP-01 (the manpower sponsor), OP-02, 03, 05 (the warfare sponsors), the Chief of Naval Material, and the fleets (for manpower documents). An additional key "user" will be the NAVMMACLANT/PAC Procedures and Analysis (P&A) teams, who will have the responsibility for reviewing and modifying the requirements documents "in house" before they are sent out for fleet review. For this P&A review, the system will produce the automated equivalent of "working papers", enabling the analyst to track the billet derivation process from inputs to outputs.

The system will include various subsystems to perform four major functions: maintain the data base, derive billet requirements related to user inputs for activities, produce manpower requirements documents (both real and hypothetical), and respond to queries regarding the data base. (It is understood here that the data base will include both input data to the billet derivation process and outputs from it.)

1.5.2 NMRS Objectives

From a user viewpoint, the objectives of the system are to perform the functions described in Section 1.5.1 in a timely, cost effective manner with a minimum of manual intervention. From a systems viewpoint, satisfying these user objectives requires the following system functions:

- Recognize a set of user commands of minimal size and translate them into the appropriate system transactions.
- Process changes to the data base in such a way as to minimize the impact of user error on the integrity/validity of the data base, and also minimize the amount of information required from the user.
- Derive the minimum (quantitative and qualitative) manpower required to support the ROC/POE as described by the user or stored in the data base; perform this derivation efficiently (so that multiple users can exercise the system without overtaxing computer/dollar resouces) and with a minimum of user input required.
- Respond to user requests in the various formats that the users will find convenient for their particular applications.
- Store and manipulate the data base in a manner that lends itself to efficient processing of a variety of users' information requirements.

1.5.3 Billet derivation (BILDER) Process

The heart of the NMRS processing is the derivation of billets in the system called BILDER which is shown in Figure 1-1. Our concern is with the SHIP central subsystem which consists of the Variability Preprocessor (Input Processor), Billet Derivation, but not Document Generation. The functional flow in the Variability Preprocessor is shown in Figure 1-2, and the Ship Billet Derivation flow is shown in Figure 1-3. In general, the billets are derived from aggregating the required watchstation information with the required maintenance manhour information to form an overall workload requirement, assigning the appropriate allowances, and applying the authorized work week to the workload by work center. It is important to note that complete watchstation information must be provided to BILDER to derive billets. The source of that information is the ROC/Watchstation Module.

1.5.4 Variability

The concept of variability in manpower involves an effort to optimize the effectiveness of the funded manpower when that level is significantly less than the documented manpower requirements. It is designed to present discrete alternatives to manpower managers from which the impact of authorizing manpower at a level below the documented level can be assessed. The scope of variability for ships influenced the design of the ROC/Watchstation Module, since the first four forms shown below deal with operational or watch-related requirements.

A. Required Operational Capability (ROC) Modification. This permits the operator to select Suboperational Capabilities to suspend from the ship's ROC and provides the manpower impact of that suspension. This alternative is actually defined in Ship Manpower Documents as Conditional Manning.

B. Suboperational Capabilities Modification. In the Ship's ROC, many Suboperational Capabilities (SOC) are identified as being provided for on a partial basis rather than a full basis. The system must be able to determine the manpower impact of changing "Partial" to "Full" for any SOC (when there is manpower impact). Additionally, where a SOC is listed "Partial" and there are other logical "Partial" conditions which should be considered, the capability to determine the manpower impact of these must be included.

C. Condition Modification. This is a modification or relaxation of certain Condition III watch requirements. This applies when it is recognized that the anticipated operational environment of the ship would not require the same degree of continuous operational readiness as wartime steaming at sea would. It provides manpower for Condition I and Condition IV watch requirements, as well as a limited Condition III capability where many selected watch stations would be manned on a two-section basis. The endurance of the ship in the modified Condition III operation would be identified in

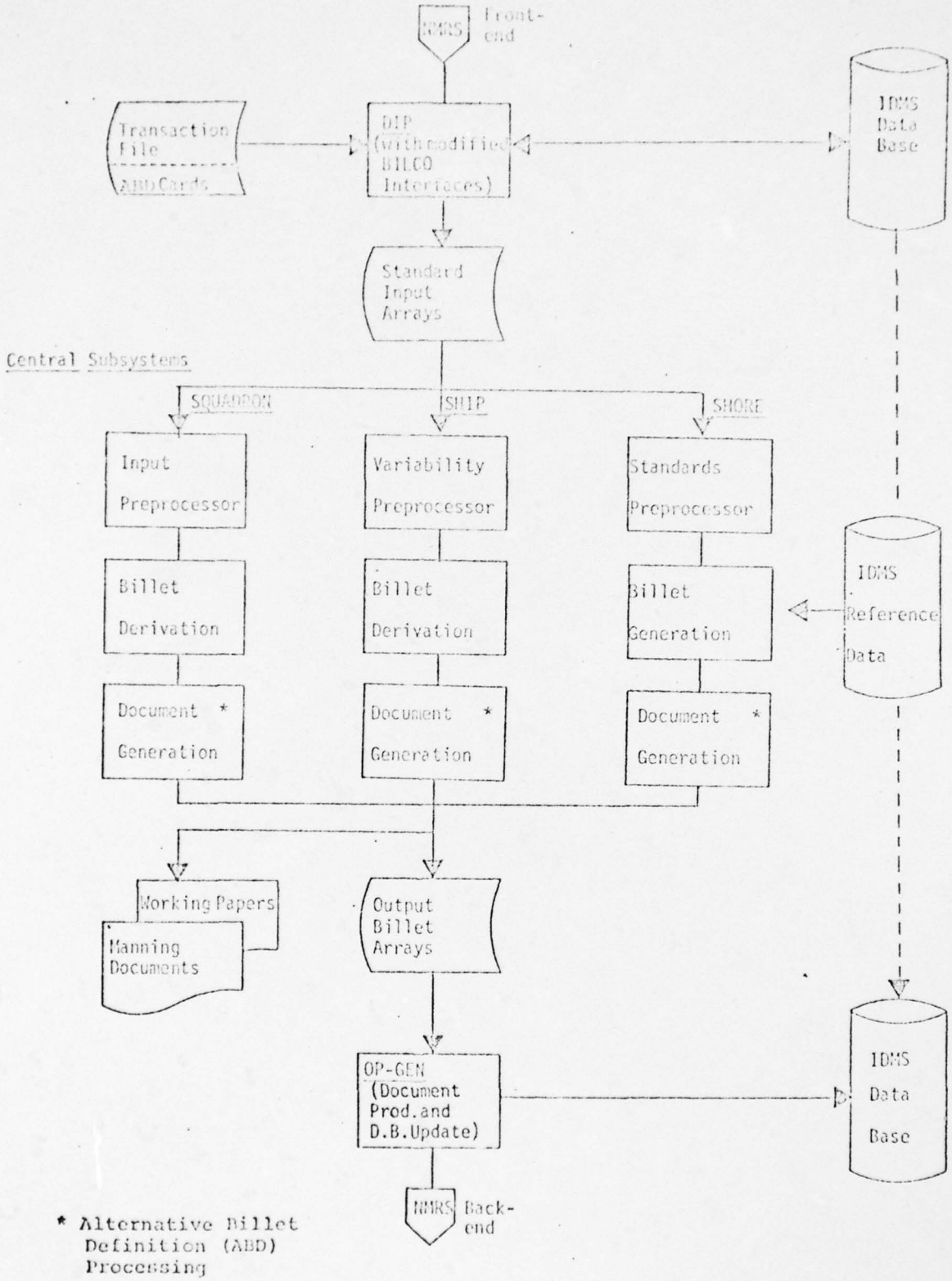


Figure 1.1 - General Approach to BILDER

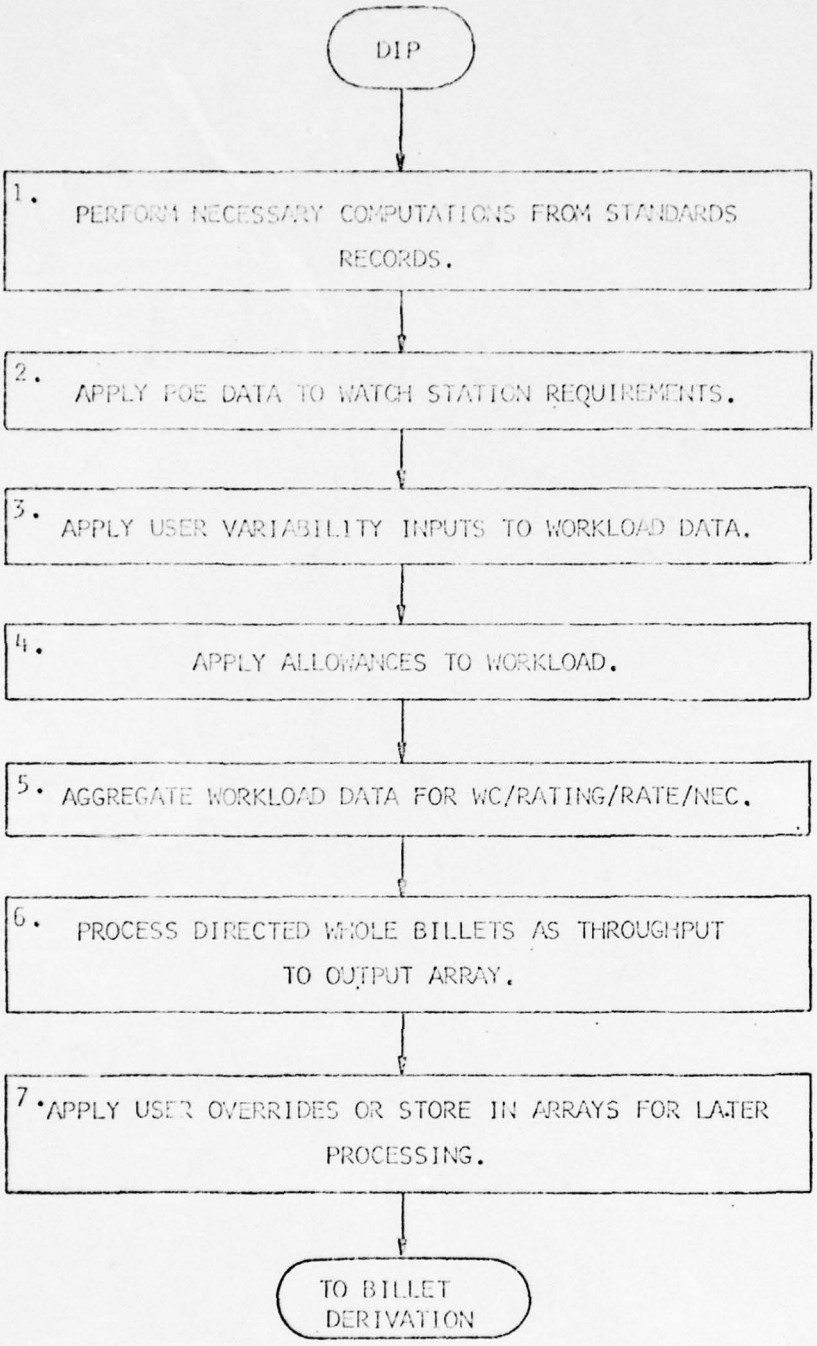


Figure 1.2 - Ship Input Processor

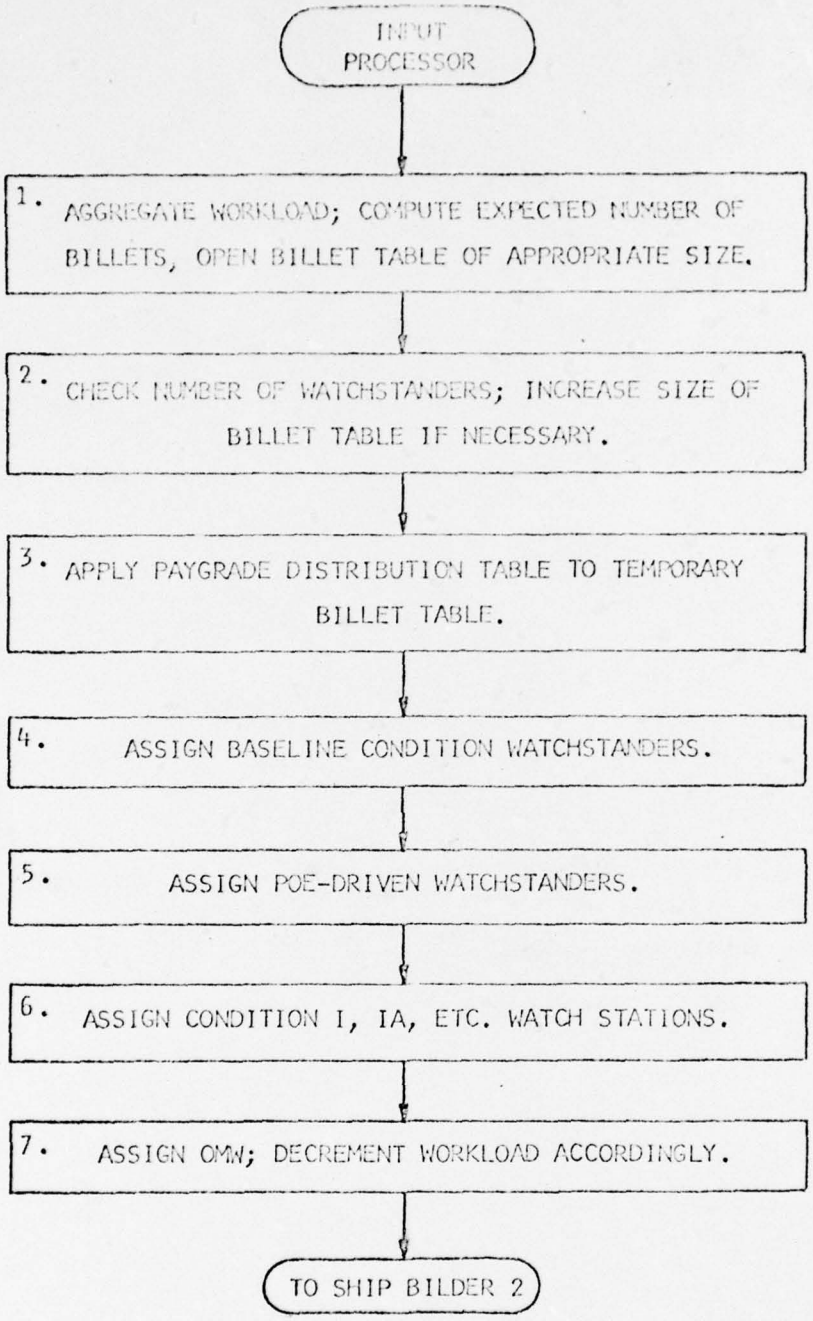


Figure 1.3 - Ship Billet Derivation

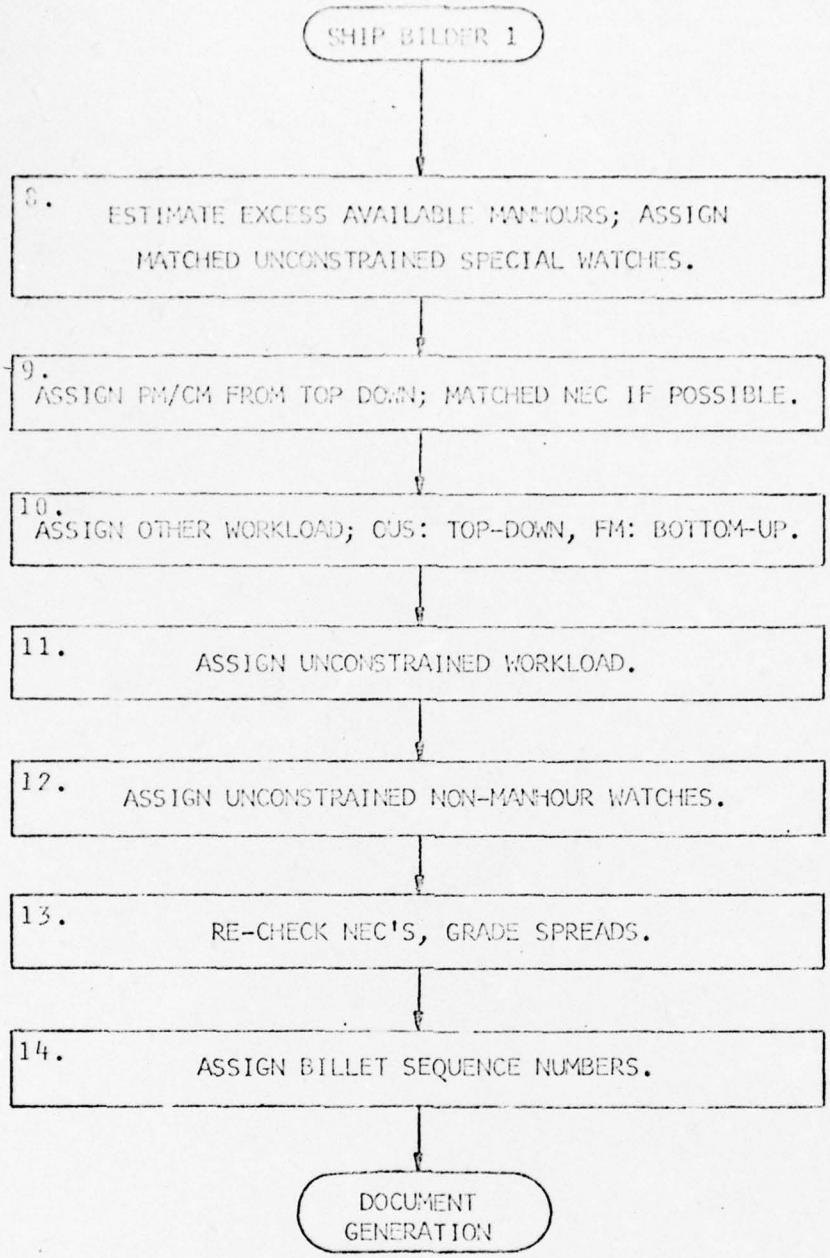


Figure 1.3 (Continued)

the Projected Operational Environment (POE). This alternative has sometimes been referred to as the "surge" capability.

D. POE Modification. This alternative applies to ships with Flight Quarters, Replenishment, or other similar requirements identified in the POE (e.g., provide UNREP services not to exceed 42 hours per week). The capability provided will enable the operator to vary the POE requirement and the system will compute the manpower impact.

E. Corrective Maintenance Modification. This would permit variations to the Corrective Maintenance input at the ship-wide or work center-wide level.

F. Work Week Modification. This permits the operator to explore the manpower impact of varying the work week length.

G. Productivity Allowance Modification. Allows variation of the Productivity Allowance factor.

1.6 Functional Requirements

Considering the above factors, functional requirements for the Ship ROC/Watchstation Module emerged as follows:

A. Given the input of a ship identification, call the ship ROC from the data base, translate the Sub-Operational Capabilities into watchstation requirements, and output to the Document Input Processor (DIP) of NMRS the minimum watch-related manpower requirements.

B. Provide an update capability so that ships not initially included in the data base can be entered by the user.

C. Provide the capability for the user to make permanent changes to ship and class data as necessary.

D. Provide the capability to enter the system with a ship identification and revised ROC, and output to DIP a revised set of minimum watch-related manpower requirements without destroying or changing the permanent data associated with that ship and class.

1.7 The Approach

In order to meet the functional requirements and operate in the environment of NMRS, the following approach was adopted:

> Compare the Ship Manpower Documents (SMD) of all available ships of a class. Note all watchstation exceptions, and where the exceptions are only rate/rating, determine if the differences are related to equipment. (Where equipments are the same, the minimum manpower requirements would be the same. In this case, the watchstations would not be added to the

exception list). The watchstation exceptions lists become the bases for the ships files, and the listings of common watchstations are the bases for the class files.

Products: (a) Listing of watchstations common to a class of ships.

(b) Listing of watchstations not common to the class, identified by ship.

Using the listings produced as described above and the manpower standards books, attach appropriate manpower information to the watchstations. Also, attach the organizational code (from the Dictionary of Organizational Codes), designator/rating, pay grade, NOBC/NEC, and watch workweek.

Product: (c) Forms which tie watchstations to minimize manpower requirements and the other data necessary for BILDER.

Develop a table which lists Sub-Operational Capabilities (SOCs) on one axis and Minor Functional Areas (i.e., pilot house, radio control, engine room, etc.) on the other. Fill in the table indicating Minor Functional Areas relationship to SOCs for a specific ship.

Product: (d) Table of SOCs vs. Minor Functional Areas.

Develop table for recording watchstations vs. SOC by Minor Functional Area. Complete the table for given ship.

Product: (e) SOC to Watchstation relationship per ship.

Using the product (c) above, develop class file and individual ship files for the ship being processed.

Product: (f) Class file input.

(g) Ship file input.

Using the exception watchstation files, develop the remainder of ship files for the class in question.

In this manner the necessary data was provided, properly formatted to the data base to permit the system to meet the functional requirements. The system functions were divided into three basic subsystems: Load, Update, and Watchstation Generation. A summary of the system is provided in Section 2 of this report and a description of the system operation is included as Section 3. The system is sub-divided into a total of seven programs or modules: NMWV01, NMWV02, NMWV03, NMWB01, NMWB02, NMWF01, and NMWF02. Descriptions of these modules, including program logic and module flow charts, are included as Appendices A through G, respectively. A description of the data base is included as Appendix H.

1.8 Results

The methodology for relating ROCs to Watchstations and extracting the data necessary to support the system were approved by NAVMMACLANT. The decision by the customer to require the use of the Cullinane Corporation Integrated Database Management System (IDMS) after the initial system design had been completed resulted in a significant delay (over one month) in the delivery of an operating system. The principal causes of the delay were training and redesign. However, the system was operational and delivered for acceptance testing within the scope of the contract.

1.9 Other Relevant Documentation

The following additional documentation was developed under this contract and delivered to the user of the system, NAVMMACLANT:

ROC/Watchstation Module Users Manual	28 May 1976
ROC/Watchstation Module Specification NMWV01	30 June 1976
ROC/Watchstation Module Specification NMWV02	30 June 1976
ROC/Watchstation Module Specification NMWV03	30 June 1976
ROC/Watchstation Module Specification NMWB01	30 June 1976
ROC/Watchstation Module Specification NMWB02	30 June 1976
ROC/Watchstation Module Specification NMWF01	30 June 1976
ROC/Watchstation Module Specification NMWF02	30 June 1976
ROC/Watchstation Module Specification Data Base	30 June 1976
ROC/Watchstation Module Specification Commands and JCL	30 June 1976

SECTION 2

SYSTEM SUMMARY

2.1 System Application

The ROC/Watchstation Module is a subsystem of the Navy Manpower Requirement System, a key element of the Navy Manpower Planning System (NAMPS). Where NMRS will produce Ship, Squadron, and Shore Manpower Documents, the ship ROC/Watchstation Module provides operational manpower information to the Document Input Processor for use in the Billet Derivation (BILDER) process for ships. Inasmuch as the ships have Required Operational Capabilities (ROCs) assigned, and manpower is provided to make the ship and its equipment operational, a natural relationship between the ROC and manpower exists. The ROC/Watchstation Module establishes this relationship in an automated system by tying Sub-operational Capabilities to watchstation requirements. Since the Projected Operational Environment (POE) for ships often contains information which impacts upon manpower requirements, it is also factored into the system where relevant. This not only provides essential inputs for document production, it also enables NMRS to assess the manpower impact of varying the operational requirements or the environment which is planned for the ship.

Additional capabilities will be developed for NAMPS at a later date to enable the aggregating of manpower requirements at various levels (i.e., ship class, ship type, rating group, program element, etc.) in order to provide manpower managers with timely information on the manpower impact of policy decisions and program changes.

2.2 System Operation

Figure 2.1-2.4 show the process flow of the subsystems of the Ship ROC/Watchstation Module. The LOAD subsystem will be operated by the maintainer of the system, the Navy Manpower and Material Analysis Center, Atlantic, anytime that new watchstation title, ROC title, class, ship, condition title, or level information needs to be entered into the data base. The outputs would include records to the data base, a display of those records, edit reports, and error reports (where relevant).

The UPDATE subsystems, also operated by NAVMMACLANT, provide the means for updating the watchstation title, ROC title, Condition title, class, ship, watchstation, watchstation ROC, level, and ROC records when necessary on an as-occurring basis. The outputs are new or changed records with a display of those records, edit reports, and error reports (where relevant).

The principle subsystem, Watchstation Generation, is operated by NAVMMACLANT when NMRS is to be run for a ship, either to produce a Ship Manpower Document or to query the system for operational manpower information. The operator input will include identification of the ship and ROC, and the output will be a watchstation file to be passed to the Document Input Processor.

LOAD Subsystem of SHIP ROC/WS System

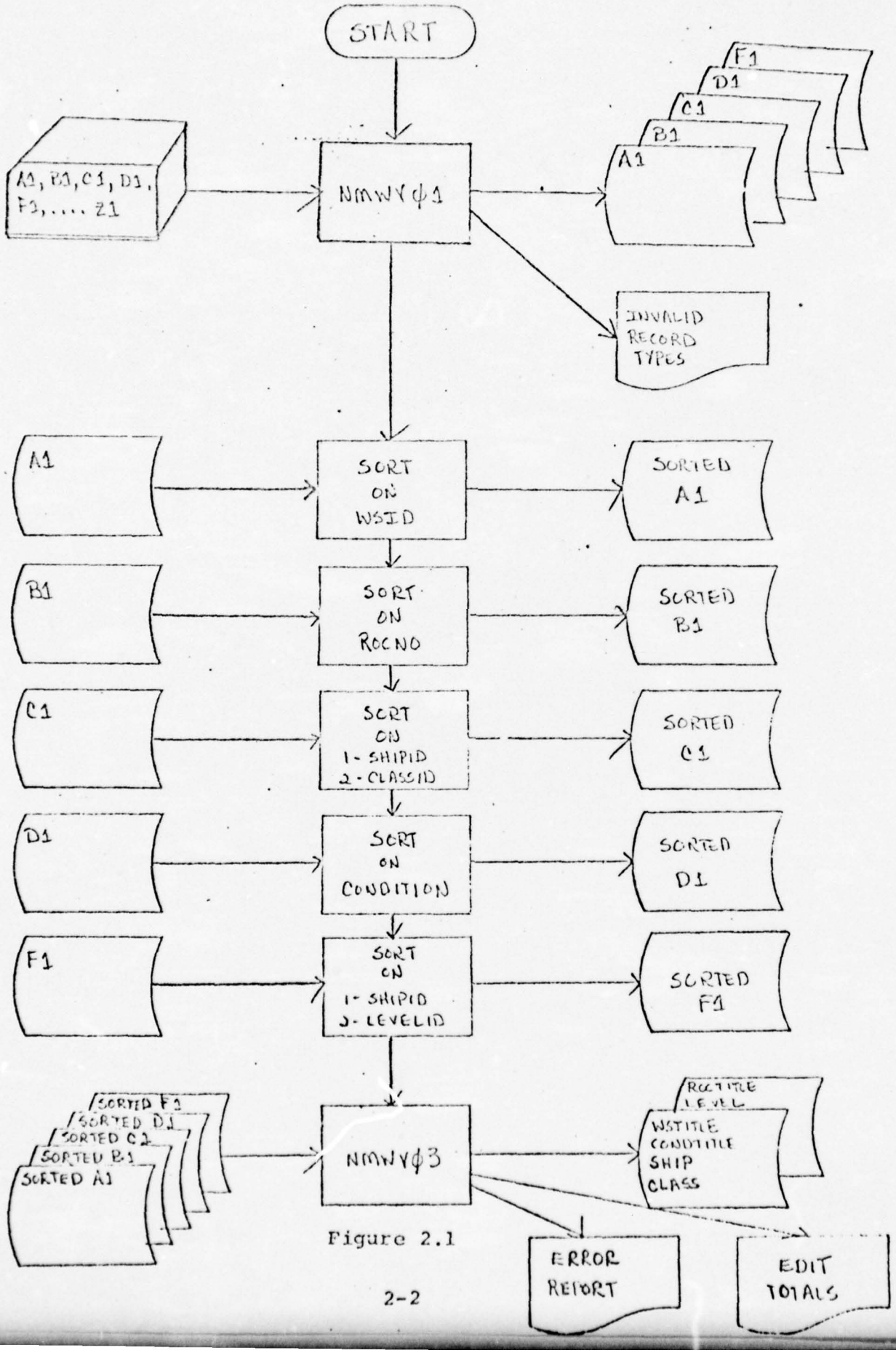


Figure 2.1

UPDATE Subsystem (1) of SHIP ROC/WS System

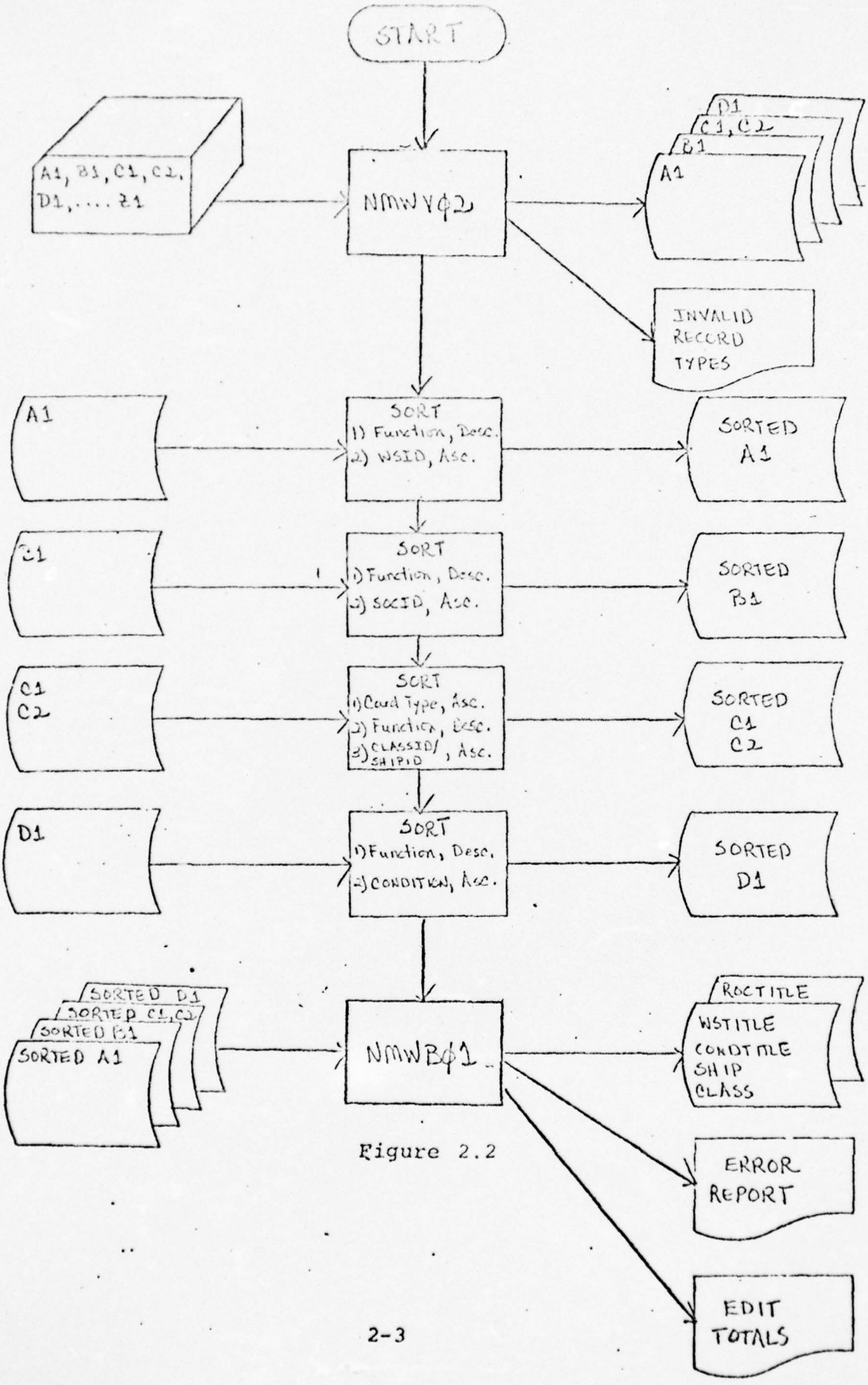


Figure 2.2

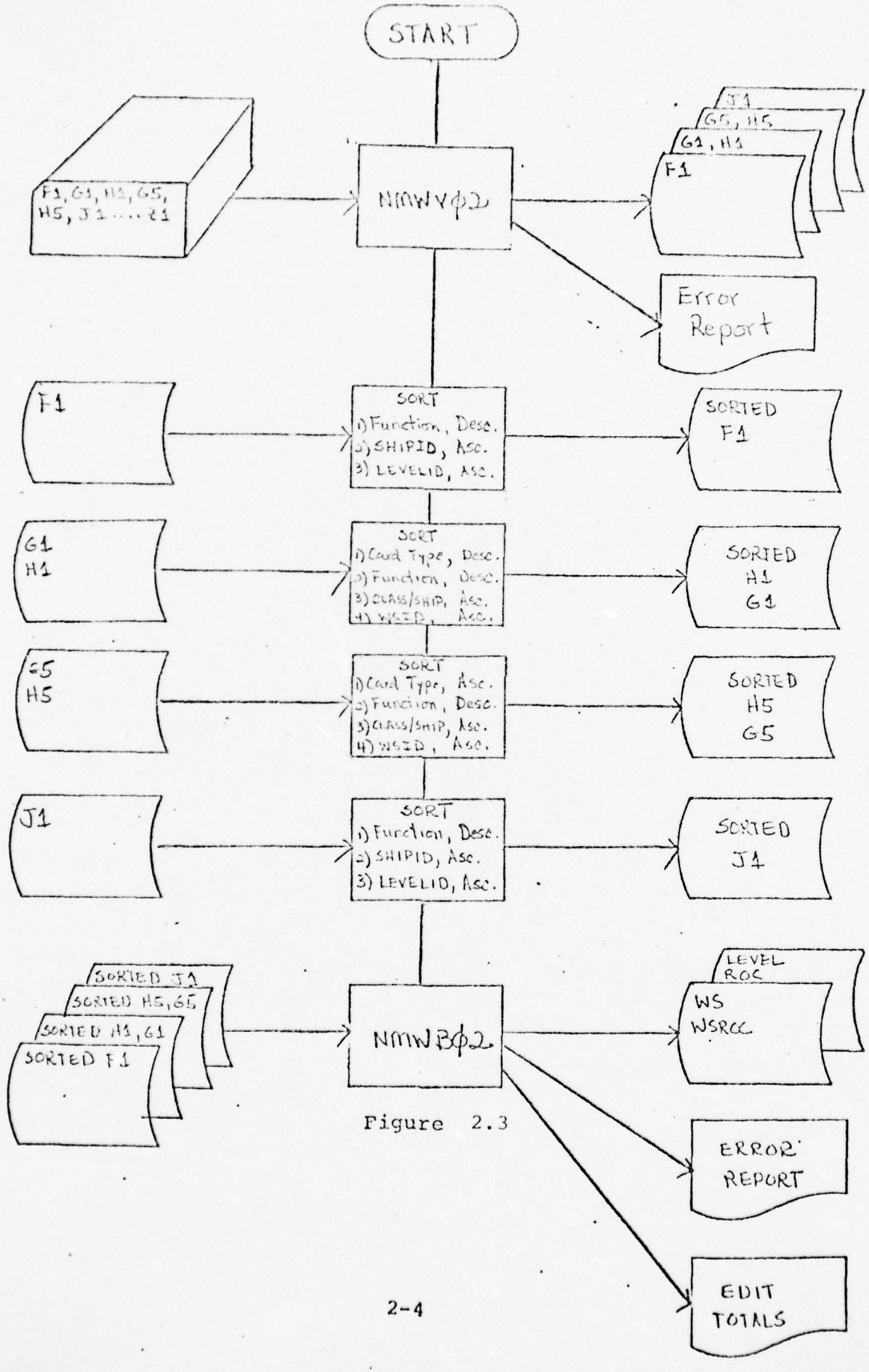

START
F1, G1, H1, G5, H5, J1 ... Z1
NMWVΦ2
J1
G5, H5
G1, H1
F1
Error Report
F1
SORT
1) Function, Desc.
2) SHIPID, Asc.
3) LEVELID, Asc.
SORTED F1
G1
H1
SORT
1) Card Type, Desc.
2) Function, Desc.
3) CLASS/SHIP, Asc.
4) WSID, Asc.
SORTED H1 G1
G5
H5
SORT
1) Card Type, Asc.
2) Function, Desc.
3) CLASS/SHIP, Asc.
4) WSID, Asc.
SORTED H5 G5
J1
SORT
1) Function, Desc.
2) SHIPID, Asc.
3) LEVELID, Asc.
SORTED J1
SORTED J1
SORTED H5, G5
SORTED H1, G1
SORTED F1
NMWBΦ2
LEVEL ROC
WS
WSROC
ERROR REPORT
EDIT TOTALS

Figure 2.3

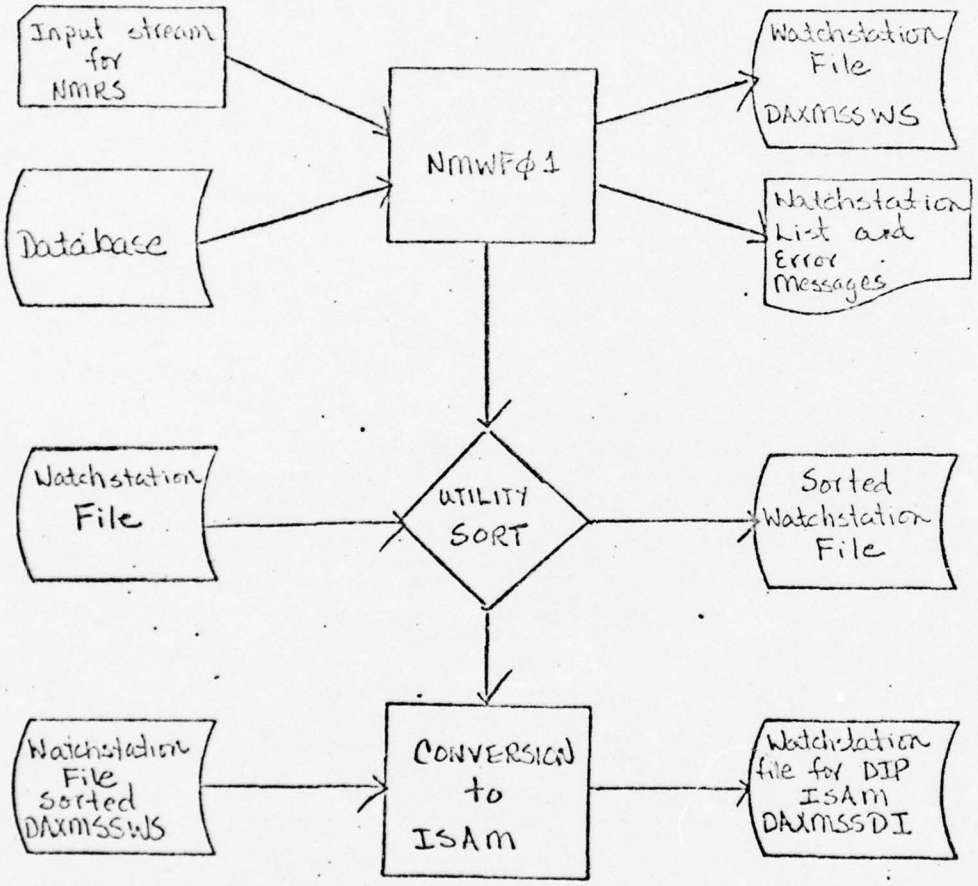

Input stream for NMRS
Database
NMWFΦ1
Watchstation File DAXMSSWS
Watchstation List and Error Messages
Watchstation File
UTILITY SORT
Sorted Watchstation File
Watchstation File sorted DAXMSSWS
CONVERSION to ISAM
Watchstation file for DIP ISAM DAXMSSDI

Figure 2.4

2.3 System Configuration

The system is currently utilizing the following equipment:

A. An IBM 370-168 computer located at NIH Computer Center.

B. Approximately one 3330 disk drive.

C. Card reader.

D. Line printer.

2.4 System Organization

The Ship ROC/Watchstation Module system may be logically divided into three subsystems: Load, Update, and Watchstation Generation. The Load subsystem edits input transactions that are submitted for the purpose of entering certain record types on the database. It uses valid transactions to load the following record types: WSTITLE, ROCTITLE, CONDTITLE, CLASS, SHIP, LEVEL. The Update subsystem is composed of two distinct run streams each of which is used to update certain record types on the database. Input transactions to this subsystem are validated before they are used to perform any operations on the database. The run stream that has as its major program NMWB01 updates the following record types: WSTITLE, ROCTITLE, CONDTITLE, CLASS, SHIP. The run stream that has as its major program NMWB02 updates these record types: LEVEL, ROC, WS, and WSROC. The Watchstation Generation Subsystem creates a minimum watchstation list for a ship and tasking level specified by an NMRS LOAD or PRODUCE command. The list is output on a file for subsequent conversion to ISAM and input to the Document Input Processor (DIP). For diagrams of the system flow for each of the Subsystems described, see Figure 2.1.

2.5 Performance

Inasmuch as the ROC/Watchstation Module is only a segment of the NMRS, performance capabilities cannot be meaningfully measured until it (NMRS) is operational with real data in the data base. However, information on inputs, outputs, and edits is presented in Section 2.7 below.

2.6 Data Base

There are nine record types on the Ship ROC/Watchstation Module Database. These record types are updated by the system and used to create a minimum watchstation list for a ship and tasking level to ultimately be used as input to DIP. The records are as follows:

A. CLASS - contains a class identifier and the UIC of the prototype ship of the class.

B. SHIP - contains UIC, shiphull and shipname.

C. WS - contains a UIC or class identifier (UIC if the watchstation is unique to this ship within its class; class code if the watchstation is common to all ships within the class); watchstation; a watchstation manning field that indicates either Marine, Lamps, CO - Manning, whether subject to surge; up to seven conditions; the following watchstander information: OEC, Rating, Associated rating, military pay grade, NEC or NOBC, number of watchstanders, hours of operational maintenance performed on watch, organizational component code, a searchkey, and a Reserve code.

D. WSROC - contains SOCID (sub-operational capability number).

E. LEVEL - contains level, UIC, level name, and update date.

F. ROC - contains SOCID

G. ROCTITLE - contains SOCID and sub-operational capability code (e.g., MOB 1.1).

F. WSTITLE - contains watchstation number and watchstation title.

G. CONDTITLE - contains condition and condition name.

2.7 General Description of Components

2.7.1 LOAD Subsystem

A. Inputs

Input will be made to the LOAD subsystem whenever there is a need to load any of the following record types onto the Database: WSTITLE, ROCTITLE, CLASS, SHIP, CONDTITLE, and LEVEL. The only control data on each input is the card type which occupies card positions one and two. The card type associates the input transaction with a particular record type; specifically A1 - WSTITLE, B1 - ROCTITLE, C1 - CLASS or SHIP, D1 - CONDTITLE, and F1 - LEVEL. Additionally, on the C1 card type there is a field which identifies the transaction as a CLASS or SHIP load transaction. The origin of the inputs will be determined by User Interface Requirements to be defined for NMRS.

B. Processing

The input transactions are first processed in NMWV01 where the card types are edited and individual output files are produced for the different card types. Any transactions with invalid card types are displayed on a Preliminary Edit Report. Also provided are totals of cards read, cards written to output files, and cards containing errors. The output files are then sorted into an order that

will ensure efficient handling in NMWV03. Once the data is input to NMWV03, each card type is edited according to data specifications related to the type record that the transaction will be used to load. If the transaction data meets the edit requirements, it will be used to load a record onto the Database. If the transaction contains invalid data it will appear on an error report followed by error message(s) that identify the field(s) in error. The entire card is edited (i.e., editing does not end when the first error is detected). Those transactions containing errors must be corrected and resubmitted in the next run. An edit report will be printed, identifying, by card type, the number of transactions read, those written (to the database), and those containing errors.

C. Outputs

The following output may be produced: An error report identifying invalid data on the input transactions; an edit report identifying by card type totals of cards read, cards used to write to the Database, and cards with errors; a display of records loaded onto the Database. Any of the record types identified in the 'Input' section may be loaded onto the Database depending on the types of transactions entered.

2.7.2 Update Subsystem (1)

A. Inputs

Input will be made to Update Subsystem (1) when any of the following record types on the Database require an update: WSTITLE, ROCTITLE, CONDTITLE, CLASS, SHIP. Each input record may be identified by information provided in card columns one through eight, which contain a Transaction Identifier (1), Sequence Number (2-3), Record Identifier (4-5) and Transaction Code (6-8). Input function is found in card column nine, and input card type is located in card columns ten and eleven. The card type associates the input transaction with a particular record type on the Database; specifically A1 - WSTITLE, B1 - ROCTITLE, C1 - CLASS, C2 - SHIP, and D1 - CONDTITLE. The origin of these inputs will be determined by User Interface Requirements to be defined for NMRS.

B. Processing

The input transactions are first processed in NMWV02 where card type and Transcode are edited and output files are generated for the different card types. Any cards with either an invalid card type or an invalid Transcode will appear on an edit report and will receive no further processing by the system. A report will be printed specifying totals for cards read, cards written to output files, and cards containing errors. Each output file is then sorted into an order that will ensure efficient processing in NMWB01, the update module. In NMWB01, the editing of each transaction is based on data specifications for the type record the transaction will update. If the transaction contains valid data, it is used to perform the operation specified by the card function (add, change, or delete). If the transaction contains invalid data, it will be printed on an error report accompanied by error messages to identify the problem fields. An edit report is printed identifying by card type and

record type the following totals: cards read, records added, records changed, records deleted, and cards with errors.

C. Outputs

The following printer outputs are produced: An error report displaying cards with invalid data; an edit report providing totals indicated in the Processing section; a display of records changed on the Database or added to the Database. Output may also include new records on the Database or changes to existing records on the Database.

2.7.3 Update Subsystem (2)

A. Inputs

Input will be made to Update Subsystem (2) when any of the following record types on the Database require an update: WS, WSROC, LEVEL, and ROC. Each input record may be identified by information provided in card columns one through eight which contain a Transaction Identifier (1), Sequence Number (2-3), Record Identifier (4-5) and Transaction Code (6-8). Input function is found in card column nine, and input card type is in card columns ten and eleven. The card type associates the input transaction with a particular record type on the Database; specifically G1 and H1 - WS, G5 and H5 - WSROC, F1 - LEVEL, and J1 - ROC. The origin of these inputs will be determined by User Interface Requirements to be defined for NMRS.

B. Processing

The input transactions are first processed in NMWV02 where card type and Transcode are edited and output files are generated for the different card types. Any cards with either an invalid card type or an invalid Transcode will appear on an edit report and will receive no further processing by the system. A report will be printed specifying totals for cards read, card written to output files, and cards containing errors. Each output file is then sorted into an order that will ensure efficient processing in NMWB02, the update module. In NMWB02, each transaction is edited based on data specifications for the type record the transaction will update. If the transaction contains valid data, it is used to perform the operation specified by the card function (add, change, or delete). If the transaction contains invalid data, it will be printed on an error report followed by error messages to identify the problem fields. An edit report is printed identifying by card type and record type the following totals: cards read, records added, records changed, records deleted, and cards with errors.

C. Outputs

The following printer outputs are produced: An error report displaying cards with invalid data; an edit report providing

totals indicated in the Processing section; a display of records modified on the Database or added to the Database. Output may also include new records on the Database or changes to existing records on the Database.

2.7.4 Watchstation Generation Subsystem

A. Inputs

The purpose of the input to the Watchstation Generation program (NMWF01) is to control the production of a watchstation file for input to DIP. The input will contain an NMRS PRODUCE or LOAD and identification for the ship and ROC for which a watchstation list is required. The Ship ROC/Watchstation Database is an associated input to this subsystem, providing organizational data for ships, ship classes, ship conditions, ROC elements, and watchstations. It also contains specific data by ship and class for watchstations and ROC's and identifies the relationships between them. The origin of the input is to be identified by User Interface Requirements to be defined for NMRS. Data Files associated with the input are the Input Transaction File which consists of NMRS transaction cards, specifically the PRODUCE and/or LOAD commands; and the Ship ROC/Watchstation Module Database.

B. Processing

Each input PRODUCE or LOAD command will generate a file of all watchstations required for the ship under its specified ROC. The command identifies the ship and the ROC. The database contains watchstations common to all ships in a class as well as the watchstations specific to each ship. Each class watchstation is first examined to determine which ROC elements drive it. Then each of these ROC elements is checked against the ship's ROC. If the ship's ROC specifies a ROC element which drives the watchstation, then the watchstation must be manned, and a watchstation record is output to the watchstation file for each watchstander identified on the WATCH record. The watchstation file is then sorted by UIC, Tasking LEVEL, and the first occurrence of ORGCODE on the record, and converted to Indexed Sequential organization for input to the Document Input Processor (DIP) Subsystem of NMRS.

C. Outputs

Output consists of a watchstation file which contains one record for each watchstander at each watchstation selected for the specified ship and ROC. The file is sorted and converted to Indexed Sequential organization in order to provide WATCH data input to NMRS for the production of Ship Manpower Documents. The output contains identification of the ship (UIC) and tasking level (LEVEL) for which the run was made, a watchstation ID number and the following watchstander requirements: Rate/Rating/NEC, or Rank/Designator/NOBC, OMW time, and organizational component listed by ship condition. Once the watchstation file is converted to ISAM, it is passed to the Document Input Processor Subsystem (DIP) for use by the SHIP process of the BILDER subsystem of NMRS.

SECTION 3

SYSTEM OPERATION

3.1 Input Requirements

3.1.1 Input to LOAD Subsystem

Input to the LOAD subsystem will be required randomly, as a function of the need to load certain record types onto the Database; specifically WSTITLE, ROCTITLE, CONDTITLE, CLASS, SHIP, and LEVEL. The input must be entered on punched cards and will be generated by a staff unit which is to be defined by User Interface Requirements for NMRS.

3.1.2 Input to Update Subsystem

Input to the Update Subsystem (1) will be required randomly, as a function of the need to update the following record types on the Database: WSTITLE, ROCTITLE, CONDTITLE, CLASS, and SHIP. Input to the Update Subsystem (2) will be required randomly, as a function of the need to update the following record types on the Database: WS, WSROC, LEVEL, and ROC. The input must be entered on punched cards and will be generated by a staff unit which is to be defined by User Interface Requirements for NMRS.

3.1.3 Input to Watchstation Generation Subsystem

Input to the Watchstation Generation Subsystem will be required randomly, as a function of the need to produce a Ship Manpower Document for the first time, or due to any changes to the ship ROC which might impact the watchstations. The input will be on a punched card and will be generated by a staff unit (Manpower Analyst) which is to be defined by User Interface Requirements for NMRS. An associated input is the Ship ROC/Watchstation Module Database.

3.2 Composition Rules

All card inputs to the system must have NMRS Header information in positions 1 - 8. This header contains a Transaction Identifier in position 1, a sequence number in positions 2-3, a record identifier in positions 4-5 and a transaction code in positions 6 - 8. The header information is given below by card type:

A1 - A0181ROC
B1 - A0183ROC
C1 - A0184ROC Update Subsystem (1)
C2 - J0185ROC
D1 - A0182ROC

F1 - M0186ROC
G1 - R0188ROC
H1 - R0188ROC Update Subsystem (2)
G5 - T0189ROC
H5 - T0189ROC
J1 - M0187ROC

A01 CMD Watchstation Generation

The sequencing of all input to the Update Subsystems is controlled by utility sorts and is sorted to optimize the efficiency of the system. The user should be aware that for a particular card type, all delete functions will be executed first, the change functions second, and the add functions third. The only functional restriction hard-coded in the system is that a CLASS record may not be deleted until all of its member SHIP records have been deleted, and these operations may not occur during the same run.

3.3 Vocabulary

The Users Manual contains valid combinations of UICs and classes and identifies the prototype ship of each class. It also contains a listing of ROC numbers and their associated sub-operational capability codes.

3.4 Input Formats

In order to prepare valid input to the system the following specifications must be met. The criteria will be given by card type and will be keyed to card columns that may be identified on the input layout forms. If an explanation for a particular field on a card is omitted; that field is not edited and any value entered will be accepted by the system. Record layout forms are included in Appendix C.

3.4.1 Load Subsystem

A. Card type A1 (cc. 1 - 2)

- Major Functional Area of the Watchstation must be numeric (cc. 3 - 5)
- Minor Functional Area of the Watchstation must be numeric or spaces (cc. 6 - 8)
- Sequence Code of the Watchstation must be numeric or blank. It must be blank if the Minor Functional Area is blank (cc. 9 - 12)
- Watchstation Title must be present (cc. 13-44)

B. Card type B1 (cc. 1 - 2)

- ROC number must be numeric (cc. 3-6)
- Mission Area must be alphabetic (cc. 7 - 9)
- Function Code must be right-justified numeric (cc. 10 - 11)

- Required Functional Capability Code must be right-justified numeric (cc. 12 - 13)

C. Card type C1 (cc. 1 - 2)

- UIC must be present (cc. 3 - 7)
- Class must be numeric (cc. 8 - 10)
- Record Identifier must be 'CLS' if a CLASS record is to be loaded; must be blank if a SHIP record is to be loaded (cc. 11 - 13)
- Shiphull must be present (cc. 14 - 23)
- Ship name must be present (cc. 24 - 55)

D. Card type D1 (cc. 1 - 2)

- Condition must be present (cc. 3 - 4)
- Condition title must be present (cc. 5 - 34)

E. Card type F1 (cc. 1 - 2)

- Level must be present (cc. 3 - 14)
- UIC must be present (cc. 15 - 19)
- Level name must be present (cc. 20 - 51)

3.4.2 Update Subsystem (1)

A. Card type A1 (cc. 10 - 11)

- Major Functional Area of the Watchstation must be numeric (cc. 12 - 14)
- Minor Functional Area of the Watchstation must be numeric or spaces (cc. 15 - 17)
- Sequence Code of the Watchstation must be numeric or blank (cc. 18 - 21). It must be blank if the Minor Functional Area is blank.
- Watchstation Title must be present (cc. 22 - 53)

B. Card type B1 (cc. 10 - 11)

- ROC number must be numeric (cc. 12 - 15)
- Mission Area must be alphabetic (cc. 16 - 18)
- Function Code must be right-justified numeric (cc. 19 - 20)
- Required Functional Capability Code must be right-justified numeric or spaces (cc. 21 - 22)

C. Card type C1 (cc. 10 - 11)

- Class must be numeric (cc. 12 - 14)
- UIC must be present (cc. 15 - 19)

D. Card type C2 (cc. 10 - 11)

- UIC must be present (cc. 12 - 16)
- Class must be numeric (cc. 17 - 19)
- Shiphull must be present (cc. 20 - 29)
- Ship name must be present (cc. 30 - 61)

E. Card type D1 (cc. 10 - 11)

- Condition must be present (cc. 12 - 13)
- Condition title must be present (cc. 14 - 43)

Note 1: If card function is 'D', only data in the NMRS Header and KEY subdivisions must be coded. (See Input Layout Forms.)

Note 2: For all card types, Transcode (cc. 6 - 8) must be 'ROC'; function (cc. 9) must be 'A', 'C', or 'D'.

3.4.3 Update Subsystem (2)

A. Card type F1 (cc. 10 - 11)

- UIC must be present (cc. 12 - 16)
- Level must be present (cc. 17 - 28)
- Level name must be present (cc. 29 - 60)
- Update date must be numeric (cc. 61 - 66)
- Card function must be 'A', 'C', or 'D' (cc. 9)

B. Card types G1 and H1 (cc. 10 - 11)

- UIC must be present on type G1 (cc. 12 - 16). The UIC must exist in the SHIP file on the Database.
- Class must be numeric and must be followed by two spaces on type H1. The class must exist in the CLASS file on the Database. (cc. 12 - 16)
- Watchstation number must be numeric and it must exist in the WSTITLE file on the database. (cc. 17 - 26)
- WSMANNING must be space, 'C', 'L', 'M', 'X', or space (cc. 27)
- Conditions must be in the CONDTITLE file on the database and must be left-justified on the input transaction. (cc. 28 - 41)
- The OEC for the watchstander must be 'O' or 'E'. (cc. 42)
- It is invalid for Rating/Designation (cc. 43 - 46) and Associated Rating (cc. 47 - 49(to both be present.
- The designator for an officer must be 'OFF', spaces, or numerics. (cc. 43 - 46)
- The pay grade for an enlisted watchstander must be one of the following: 'AR', 'SR', 'FR', 'AA', 'SA', 'FA', 'AN', 'SN', 'FN', '3Δ', '2Δ', '1Δ', 'CΔ', 'CS', 'CM'. (cc. 50 - 51)
- The pay grade for an officer must be in the range 'C' - 'P' (cc. 50 - 51)

- The number of watchstanders field must be in the range '01' - '09', or spaces. (cc. 56 - 57)
- The OMW per week field must be numeric and has an assumed decimal (99v99). (cc. 58 - 61)
- Either Search key (cc. 67-70) or Org. Code (cc. 62 - 66) must be present.
- If Search key is present, it must be numeric (cc. 67 - 70)
- Card function must be 'A', 'C', or 'D' (cc. 9)

C. Card types G5 and H5 (cc. 10 - 11)

- UIC must be present on G5 transaction and must exist in the SHIP file on the database. (cc. 12 - 16)
- Class must be numeric on the H5 transaction and must exist in the CLASS file on the database. It must be followed by two spaces. (cc. 12 - 16)
- Watchstation number must be numeric and it must exist in the WSTITLE file on the database. (cc. 17 - 26)
- The ROCs must be numeric and left-justified on the input transaction. They must exist in the ROCTITLE file on the database. Leading zeroes must be coded. (cc. 28 - 43, 45 - 60, 62 - 77)
- Card function must be 'A' or 'D'. (cc. 9)

D. Card type J1 (cc. 10 - 11)

- UIC must be present and must exist on the SHIP file on the database. (cc. 12 - 16)
- Level must be present. (cc. 17 - 28)
- Card function must be 'A' or 'D'. (cc. 9)
- ROCs must be numeric and left-justified on the input transaction. They must exist on the ROCTITLE file on the database. Leading zeroes must be coded. (cc. 30 - 45, 47 - 62, 64 - 79).

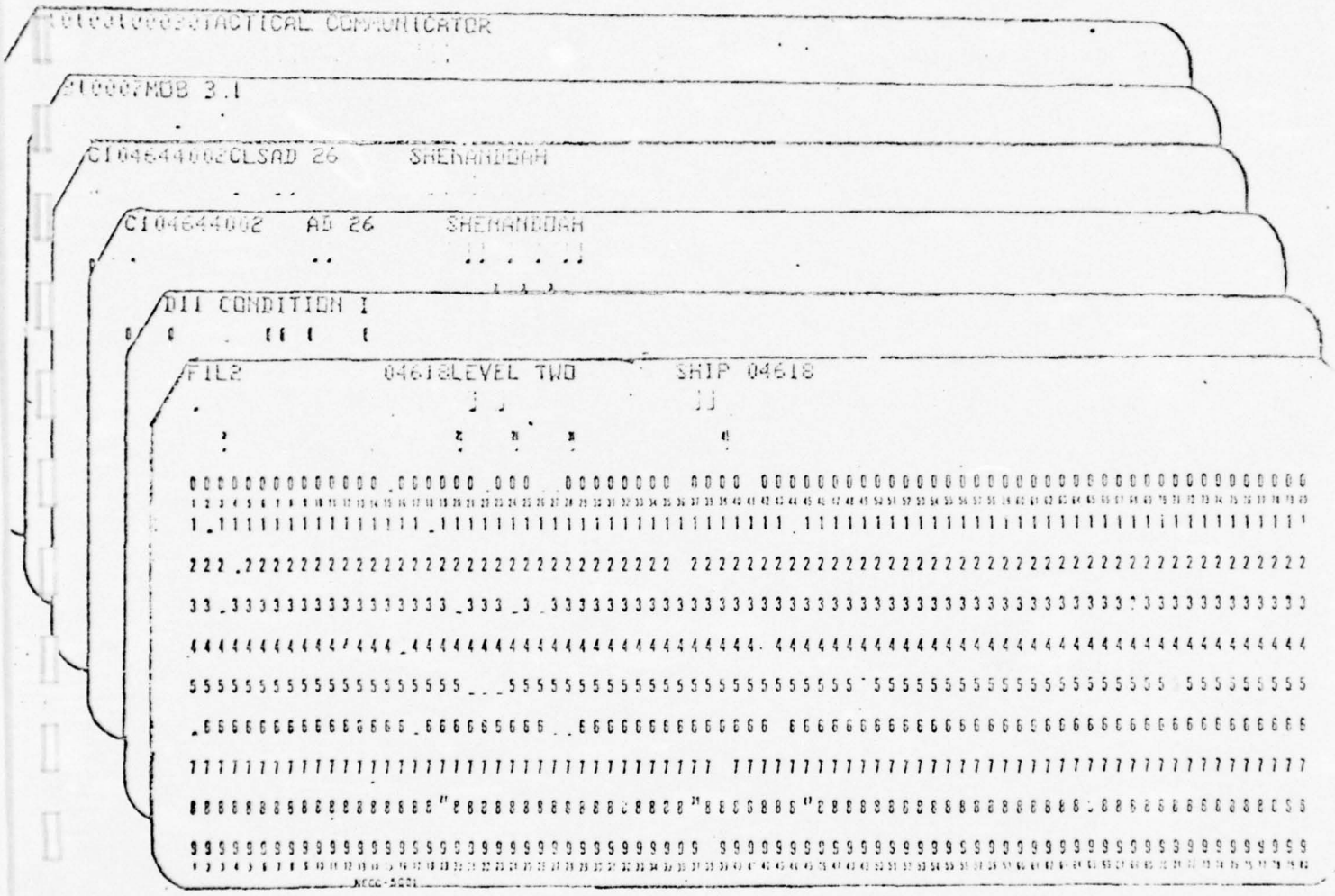
3.4.4 Watchstation Generation Subsystem

The input card has a Transaction ID in position 1 which may be anything. The sequence number in positions 2 - 3 must be '01'. The Record ID in positions 4 - 5 may be anything. The Transaction code in positions 6 - 8 must be 'CMD'. Positions 9 - 11 will contain a command code of either 'PRD' or 'LOD'. Positions 12 - 16 must contain a UIC that exists on the Database. The Tasking Level field, positions 17 - 28 must contain a tasking level that exists for the specified activity on the Database.

3.5 Sample Inputs

The following sample inputs are displayed by subsystem. The explanations will refer to the samples by card type and function, where appropriate.

3.5.1 Load Subsystem



A. Card type A1

The 'A1' indicates to the system that the card will be used to load a WSTITLE record on the Database. Card columns 3 - 12 contain the watchstation ID and card columns 13 - 44 contain the watchstation's title.

B. Card type B1

The 'B1' indicates to the system that the card data will be used to load a ROCTITLE record. Card columns 3 - 6 contain the ROC number and card columns 7 - 13 contain the code associated with that number.

C. Card type C1 (1)

The 'C1' indicates that the card data will be used to load either a CLASS record or a SHIP record. The 'CLS' in card columns 11 - 13 identify this card as one to be used to load a CLASS record. The UIC in card columns 3 - 7 is that of the prototype ship of the class. The class code is in card columns 8 - 10; the shiphull is in card columns 14 - 23; and the ship name is in card columns 24 - 55.

D. Card type C1 (2)

The second card type C1 is used to load a SHIP record. The information on the sample is identical to that of the first example of the C1 card with the exception of card columns 11 - 13. The fact that this field is blank identifies this card as one to be used to load a SHIP record.

E. Card type D1

The 'D1' indicates that this card is to be used to load a CONDTITLE record. The condition is located in card columns 3 - 4 and the condition name is in card columns 5 - 34.

F. Card type F1

The 'F1' indicates that this card is to be used to load a LEVEL record. The level identifier is located in card columns 3 - 14. The UIC to which the level is to belong is in card columns 15 - 19. The name assigned to the level appears in card columns 20 - 51.

3.5.2 Update Subsystem (1)

In the above sample inputs, the following fields will remain constant for each card type: The sequence number in card columns 2 - 3 will always be '01'; the transaction code in card columns 6 - 8 will always be 'ROC'. The transcode identifies the major system to which the card will be input. The Trans ID (cc.1) will be used for sort purposes and has no other significance. For a visual display of the delete function for each card type, see Figure 3.1.

A. Card type A1

Two fields on the card identify it as a transaction to be used to update a WSTITLE record; specifically Record ID in card columns 4 - 5 and card type in card columns 10 - 11. The first card displayed will be used to delete a WSTITLE record as indicated by the 'D' in card column 9. The only data required for this function is the watchstation ID in card columns 12 - 21. The second example of card type A1 will be used to change a WSTITLE record on the Database as indicated by the 'C' in card column 9. The watchstation ID (cc. 12 - 21) is used to identify the record to be changed. The watchstation title (cc. 22 - 53) will replace the one currently on the database for the watchstation ID specified. The third type 'A1' card has an 'A' in card column 9 and will be used to add a WSTITLE record to the Database. The watchstation ID must be unique (the card will be rejected if the watchstation ID coded already exists in a WSTITLE record) and the watchstation title must be present.

B. Card type B1

Record ID '83' and card type 'B1' identify this transaction as one to be used to update a ROCTITLE record. The first type B1 card has a 'D' in card column 9 and will be used to delete the ROCTITLE record with ROC number '0069'. The only data that must be coded for this function is ROC number. The second B1 card shown has a 'C' in card column 9 and will be used to change a ROCTITLE record. The ROC number (cc. 12 - 15) is used to identify the record to be changed. the ROC Code (Mission Area, Function Code, and Required Functional Capability Code, cc. 16 - 22) will replace what currently exists in the Database for the specified ROC number. The third B1 card has an 'A' in card column 9 and will be used to add a ROCTITLE record to the Database. The ROC number must be unique (cannot already exist on the Database) and the ROC code must be present for this function.

C. Card type C1

Record ID '84' and card type 'C1' identify this transaction as one to be used to update a CLASS record. The first C1 card will be used to delete the CLASS record with Class ID '025'. The class ID field (cc. 12 - 14) is the only data field that must be coded for the delete function. A CLASS record may not be deleted until all of the SHIP records that belong to that class have been deleted. The second C1 card displayed has a 'C' in card column 9 and will therefore be used to change a CLASS record. The class ID (cc. 12 - 14) is used to identify the record to be changed. The UIC (cc. 15 - 19) will replace the one that currently exists in the Database for the specified CLASS record. The third C1 card will add a CLASS record to the Database due to the 'A' in card column 9. The Class ID must be unique and the UIC will be that of the prototype ship of the Class.

D. Card type C2

This card type will update a SHIP record as indicated by Record ID '85' and card type 'C2'. The first type C2 card has a 'D' in card column 9 and will be used to delete the SHIP record with UIC (cc. 12 - 16) '08301'. The Class field (cc. 17 - 19) indicates the class of which the ship is a member and must be present for this function as well as the UIC. The second C2 card shown has a 'C' in card column 9 and will be used to modify a SHIP record. The UIC (cc. 12 - 16) will be used to identify the record to be changed, and the SHIP ID (cc. 17 - 19) designates the Class to which that ship belongs. The shiphull (cc. 20 - 29) and ship name (cc. 30 - 61) will replace what currently exists in those fields on the Database for the SHIP record with UIC '08301'. The third C2 card shown will add a SHIP record to the Database. The ship will become a member of the Class specified in card columns 17 - 19 and will have the shiphull and ship name indicated by those fields on the card.

E. Card type D1

Record ID '82' and card type 'D1' indicate that this transaction will be used to update a CONDTITLE record on the Database. The first D1 card has a 'D' in card column 9 and will be used to delete condition (cc. 12 - 13) 'C ' from the CONDTITLE file. The condition ID is the only data that must be coded for this function. The second example of card type D1 will be used to change a CONDTITLE record as indicated by the 'C' in card column 9. The condition (cc. 12 - 13) is used to identify the record to be changed. The condition title (cc. 14 - 43) will replace the title currently in the Database for condition '4 '. The third D1 card displayed will add a CONDTITLE record to the Database. The condition must be unique and the condition title must be present.

Card Type A1, Function 'D'

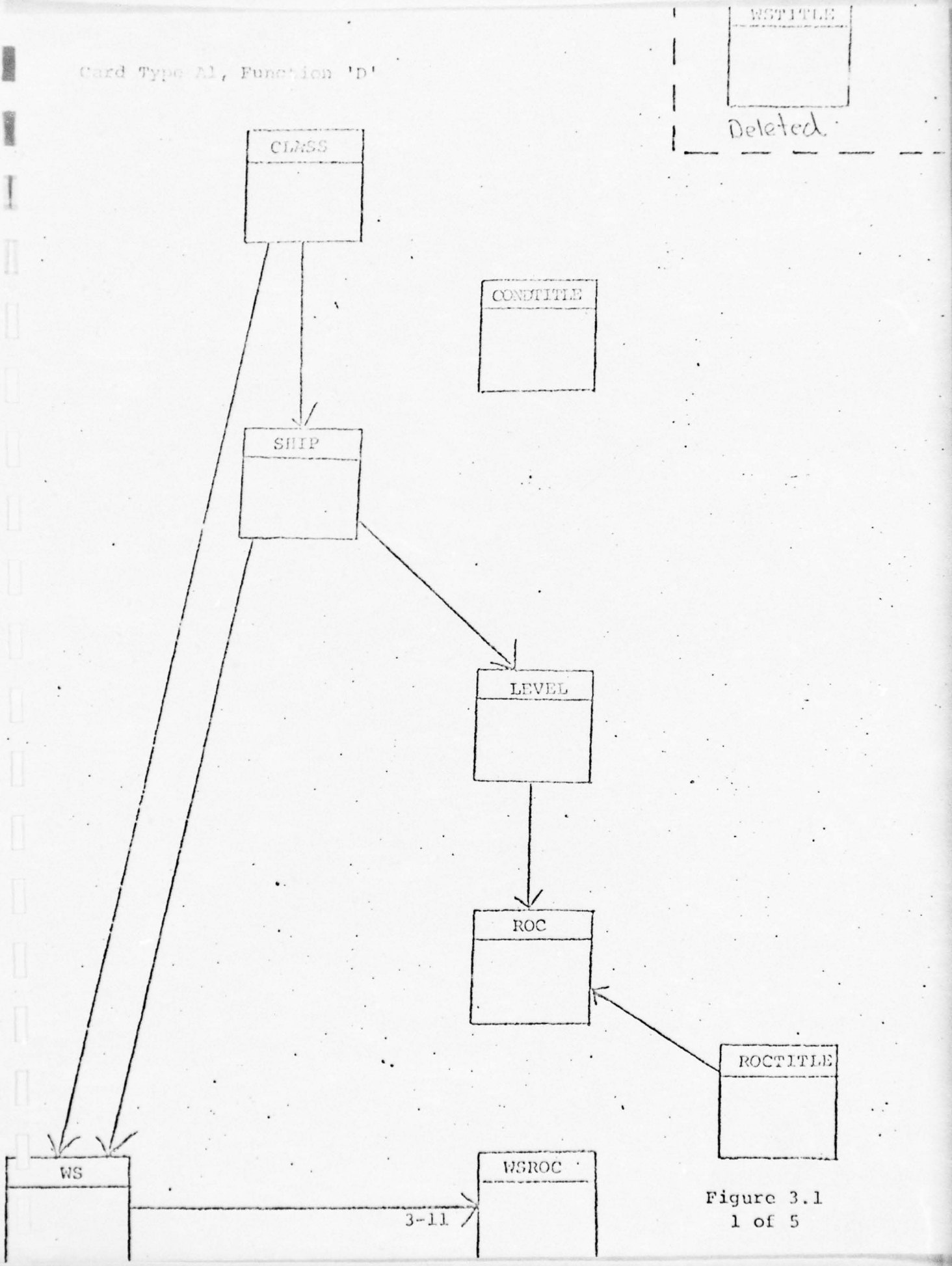


Figure 3.1
1 of 5

Card Type B1, Function D

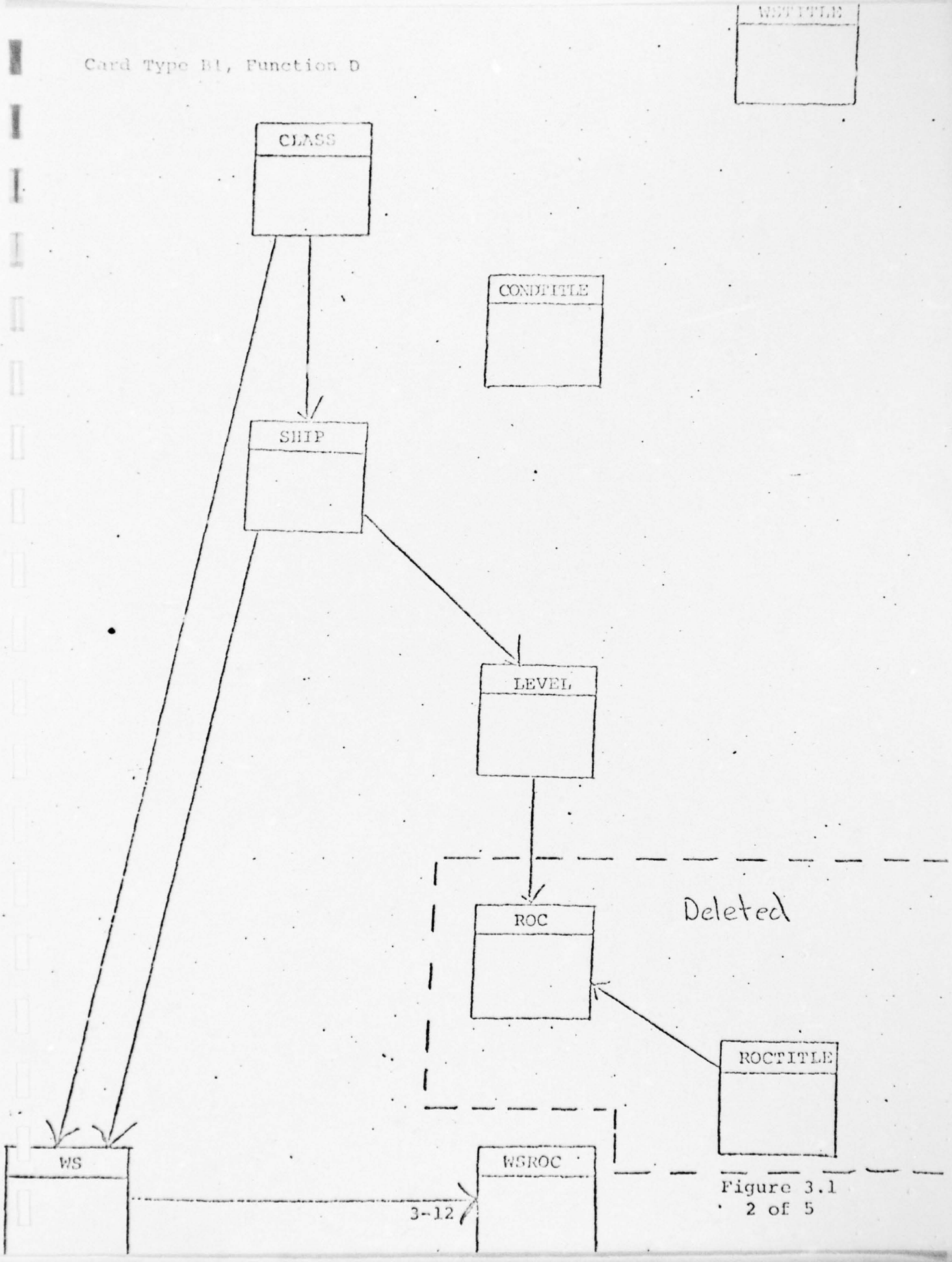


Figure 3.1
2 of 5

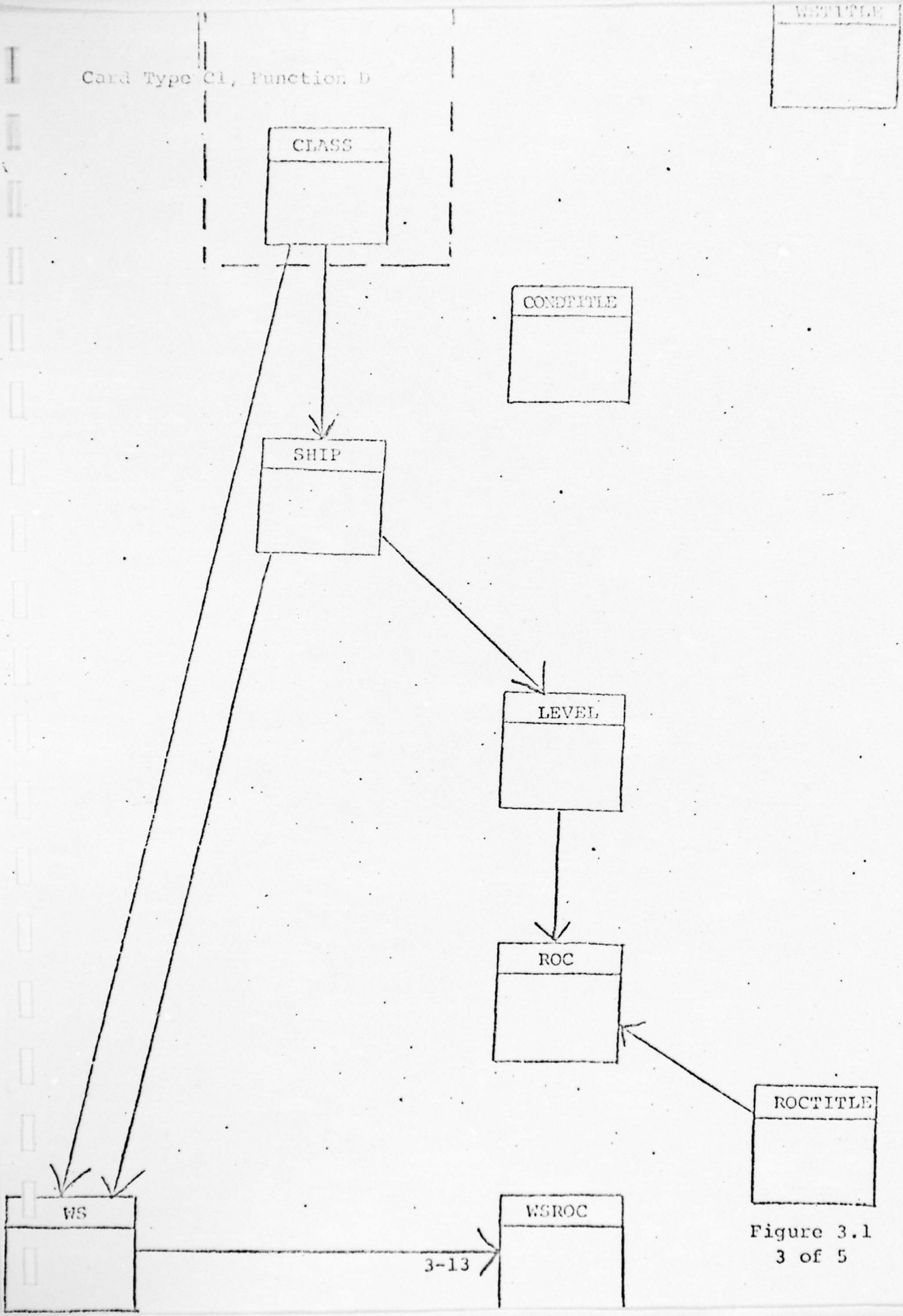


Figure 3.1
3 of 5

Card Type C2, Function D

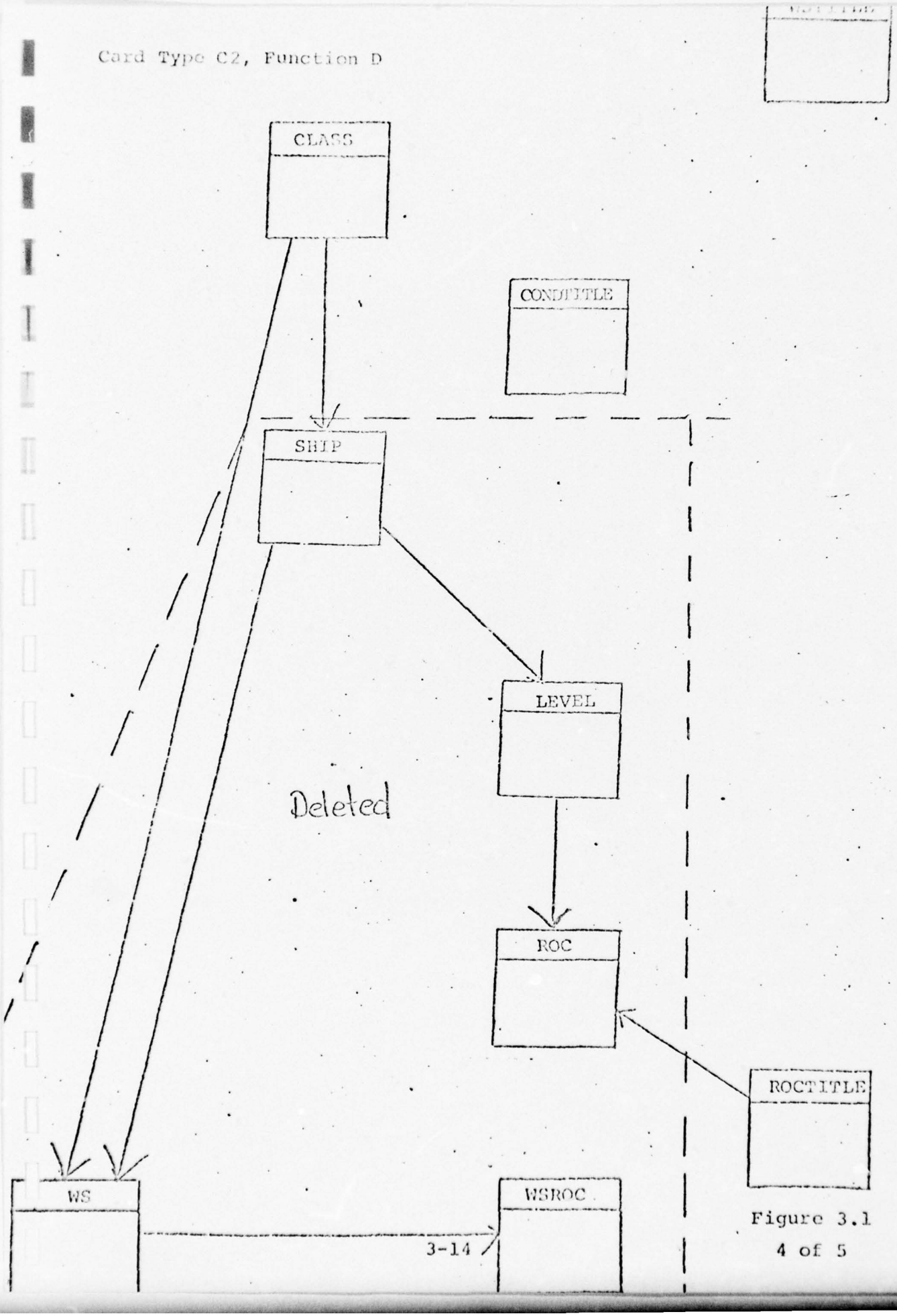


Figure 3.1
4 of 5

Card Type D1, Function D

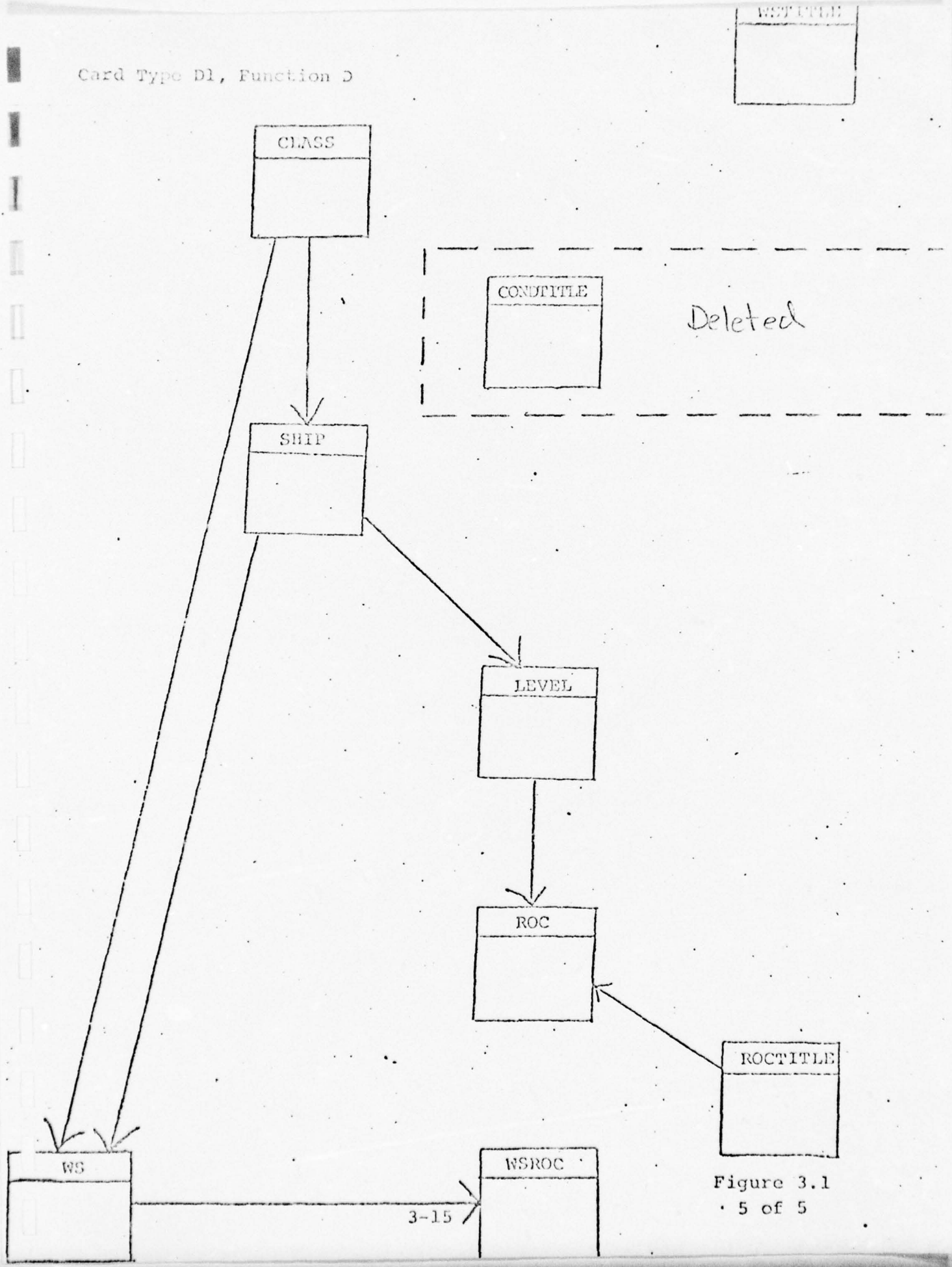
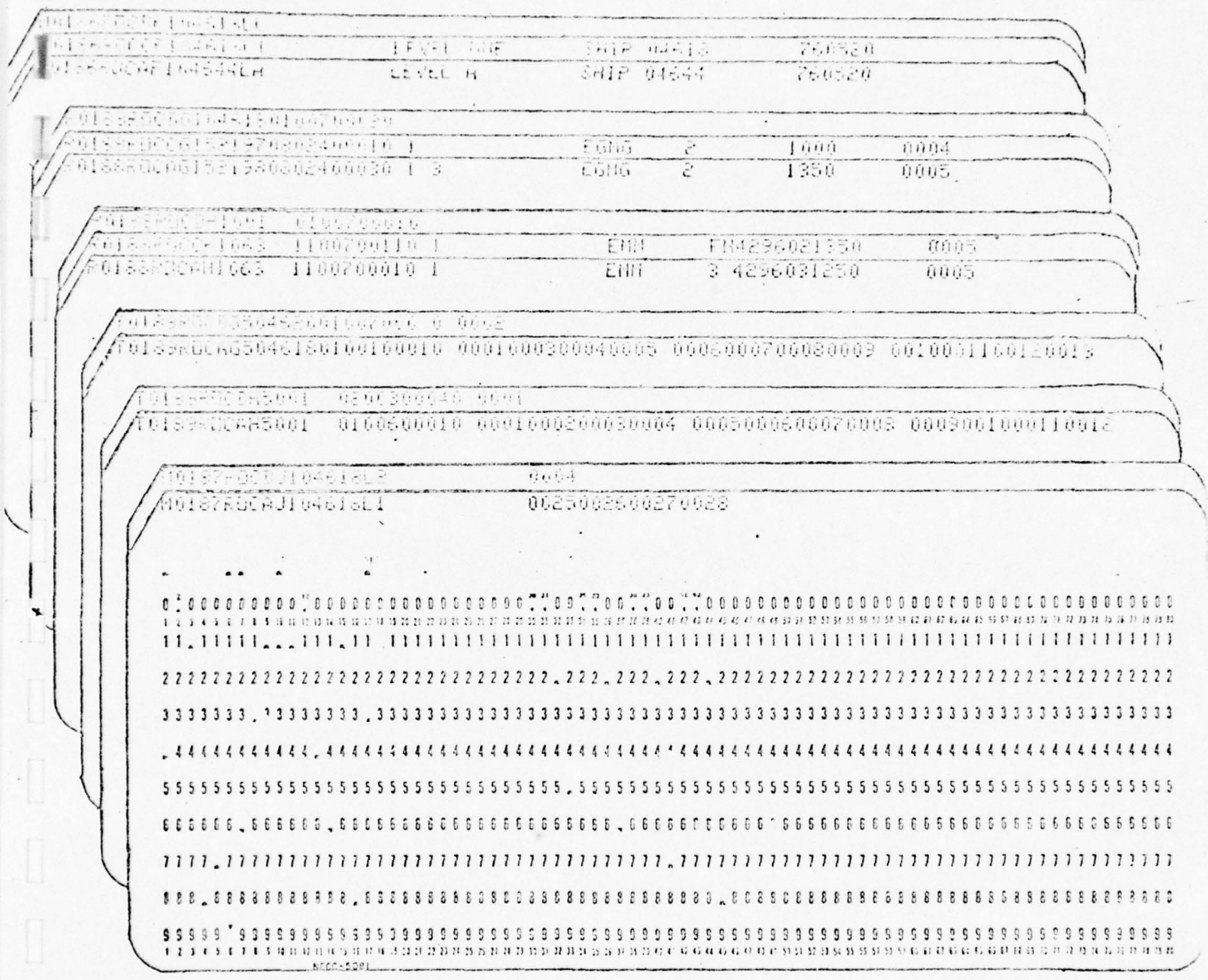


Figure 3.1
5 of 5



In the above sample inputs, the following fields will remain constant among card type: the sequence number in card columns 2 - 3 will always be '01'; the transaction code in card columns 6 - 8 will always be 'ROC'. The transcode identifies the major system to which the card will be input. the Trans ID (cc.1) will be used for sort purposes and has no other significance. For a visual display of the delete function for each card type, see Figure 3.2

A. Card type F1

This transaction is identified as one to update a LEVEL record by the '86' in the Record ID (cc. 4 - 5) and the 'F1' in card type (cc. 10 - 11). The first F1 card displayed has a 'D' in the function field (cc. 9) and will be used to delete a LEVEL record from the Database. The level ID (cc. 17 - 28) and UIC (cc. 12 - 16) identify the record to be deleted. These are the only data fields required for the delete function. The second F1 example has a 'C' in cc. 9 which indicates the change function. Again the level ID and UIC identify the record to be modified. the level name (cc. 29 - 60) and update date (cc. 61 - 66) will replace the fields that currently exist in the Database for level L1 of the Ship identified by UIC '04618'. The third F1 card displays a transaction to be used to add a LEVEL record to the Database ('A' in cc. 9). Level 'LA' will be added to the Ship with UIC '04644'. The level name is in card columns 29 - 60 and the update date is in card columns 61 - 66.

B. Card types G1 and H1

These card types have the same Record ID '88' because both are used to update a WS record. Card type H1 updates WS records that are common to all of the ships of a particular class (cc. 12 - 14). Card type G1 updates WS records that are unique for a particular UIC (cc. 12 - 16). The UIC and Class occupy the same field on the G1 and H1 card, respectively. The class on the H1 card must be followed by two spaces. Each function will be described for only one of the two card types. The first G1 example has a 'D' in card column 9 and will be used to delete a WS record from the Database. The UIC field (cc. 12 - 16) and Watchstation ID (cc. 17 - 26) are used to identify the record to be deleted. These are the only data fields that are required for this function. The second example of an H1 card has a 'C' in card column 9 and will be used to modify a WS record. The class field (cc. 12 - 14) and Watchstation ID (cc. 17 - 26) are used to identify the record to be changed. The data fields will replace the fields that currently exist for this particular Watchstation of the specified class. WSMANNING (cc. 26) is blank indicating a surgeable condition. The conditions under which the watch will be stood (cc. 28 - 41) indicate that it will only be manned for Condition I. The Watchstander requirements are in card columns 42 - 72. The Watchstander for this Watchstation will be an enlisted man ('E' in cc. 42) with a Rating Designation (cc. 43 - 46) of 'GMG', therefore no Associated Rating (cc. 47 - 49) is present. The watchstander will have a pay grade (cc. 50 - 51) of 'FN' which will be converted to a numeric code before it is entered on the Database. The watchstander's NEC (cc. 52 - 55) is '4296'. The number of watchstanders required (cc. 56 - 57) is '02'. The OMW per week (cc. 58 - 61) that may be done on watch is 13.50 hours. The organizational code (cc. 62 - 66) is blank because the Search Key (cc. 67 - 70) is present. Reserve code (cc. 71 - 72) is blank. The third example of a type G1 card has an 'A' in card column 9 and will be used to add a WS record to the Database. The watchstation (cc. 17 - 26) is one that is unique to the UIC '52198' (cc. 12 - 16).

The WSMANNING field (cc. 27) is blank, indicating that the watchstation is surgeable. Conditions I and II (cc. 28 - 41) are those for which the watch will be stood. The watchstander will be an enlisted man ('E' in cc. 42) with Rating Designation 'GMG' (cc. 43 - 46). Since Rating is present, the Associated rating field (cc. 47 - 49) must be blank. The pay grade is '2 ' (cc. 50 - 51) and will be converted to a numeric code before placement on the Database. The NEC field (cc. 52 - 55) is blank. The number of watchstanders field (cc. 56 - 57) is blank, indicating a value of '01' for that field in the Database. The OMW per week (cc. 58 - 61) is 13.50 hours. The organizational code (cc. 62 - 66) is blank becuase the Search Key (cc. 67 - 70) is present. Reserve code (cc. 71 - 72) is blank.

C. Card types G5 and H5

These card types have the same Record ID '89' because both are used to update WSROC records. Card type H5 updates WSROC records that are associated with watchstations that are common to all of the ships of a Class (cc. 12 - 14). Card type G5 updates WSROC records that are associated with a watchstation that is unique for a particular UIC (cc. 12 - 16). The UIC and class ID occupy the same field on the G5 and H5 card, respectively. The Class ID on the H5 card must be followed by two spaces. Each function will be described for only one of the two card types. The first H5 example has a 'D' in card column 9 and will be used to delete a WSROC record (may delete up to 12 records with one card). The UIC field (cc. 12 - 16) and watchstation ID (cc. 17 - 26) are used to locate the WSROC record(s) to be deleted. The only WSROC that will be deleted by this card is '0002 (cc. 28 - 31). Up to 12 codes may be present as in the format shown on the second example of the H5 card. The change function is invalid for both the G5 and H5 card. The second example of the H5 card will be used to add 12 WSROC records to the Database. The class ID (cc. 12 - 14) and watchstation ID (cc. 17 - 26) are used to indicate to the system where the new WSROC records will be placed and with what records the WSROCs will be associated. The WSROC records that will be added will have the following values: 0001, 0002, 0003, 0004, 0005, 0006, 0007, 0008, 0009, 0010, 0011, and 0012. Please note that these values were picked for use as test data only.

D. Card type J1

This card has a Record ID of '87'. The first example shown will delete one ROC record from the Database. The UIC (cc. 12 - 16) and Level ID (cc. 17 - 28) are used to identify the ship and level with which the ROC record(s) to be deleted are associated. The ROC with value '0064' will be deleted (cc. 30 - 33). Up to 12 ROC records may be deleted by using one card. The change function is invalid for this card type. The second example has an 'A' in card column 9 and will add 4 ROC records to the Database. The records will be associated with the UIC (cc. 12 - 16) and Level

(cc. 17 - 28) identified by the card. The ROC records added will have the following values: '0025', '0026', '0027', and '0028'. Up to 12 ROC records may be added using one card.

Card Type T1, Function 'D'

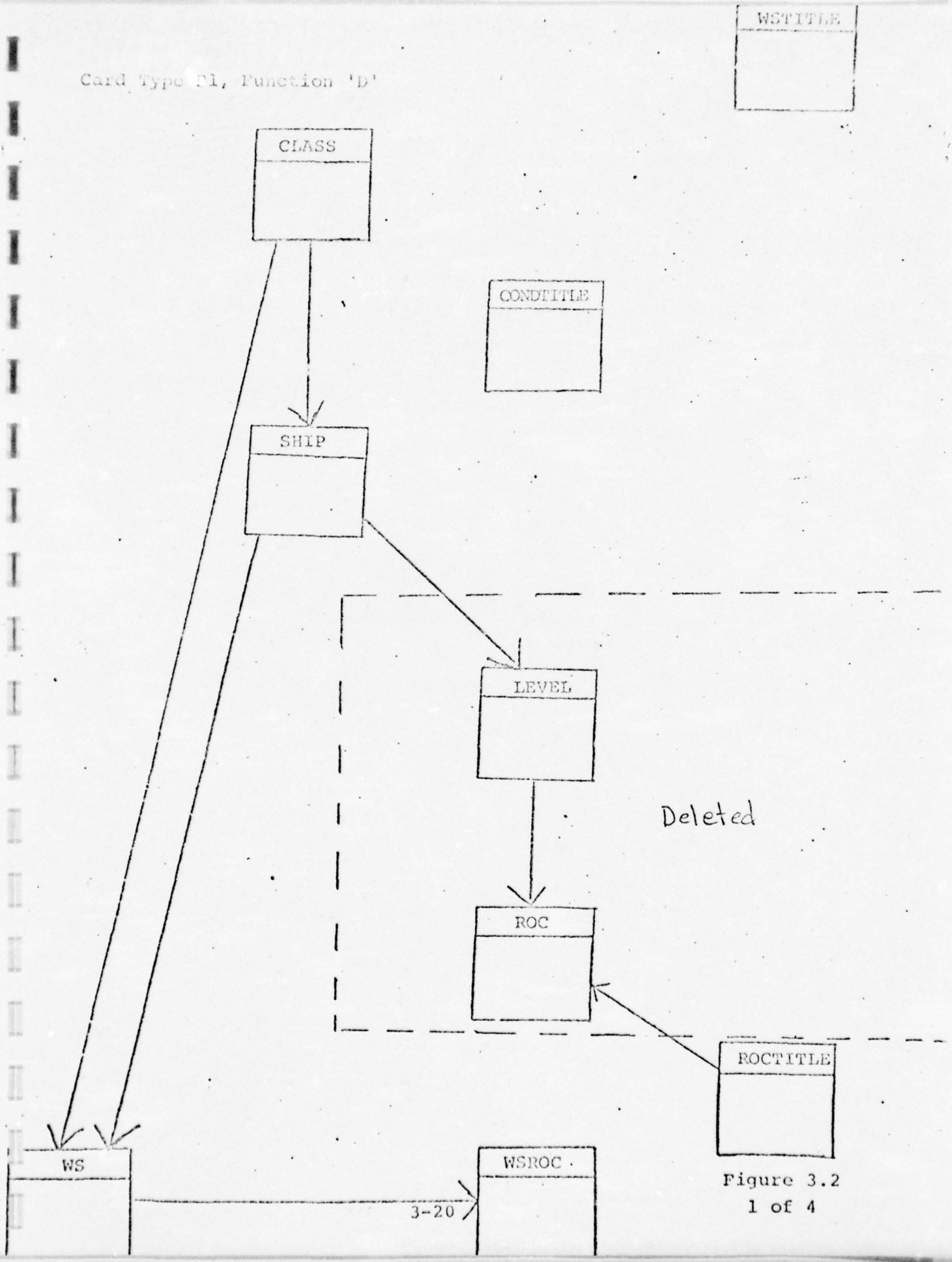


Figure 3.2
1 of 4

Card Type C1 and H1, Function 'D'

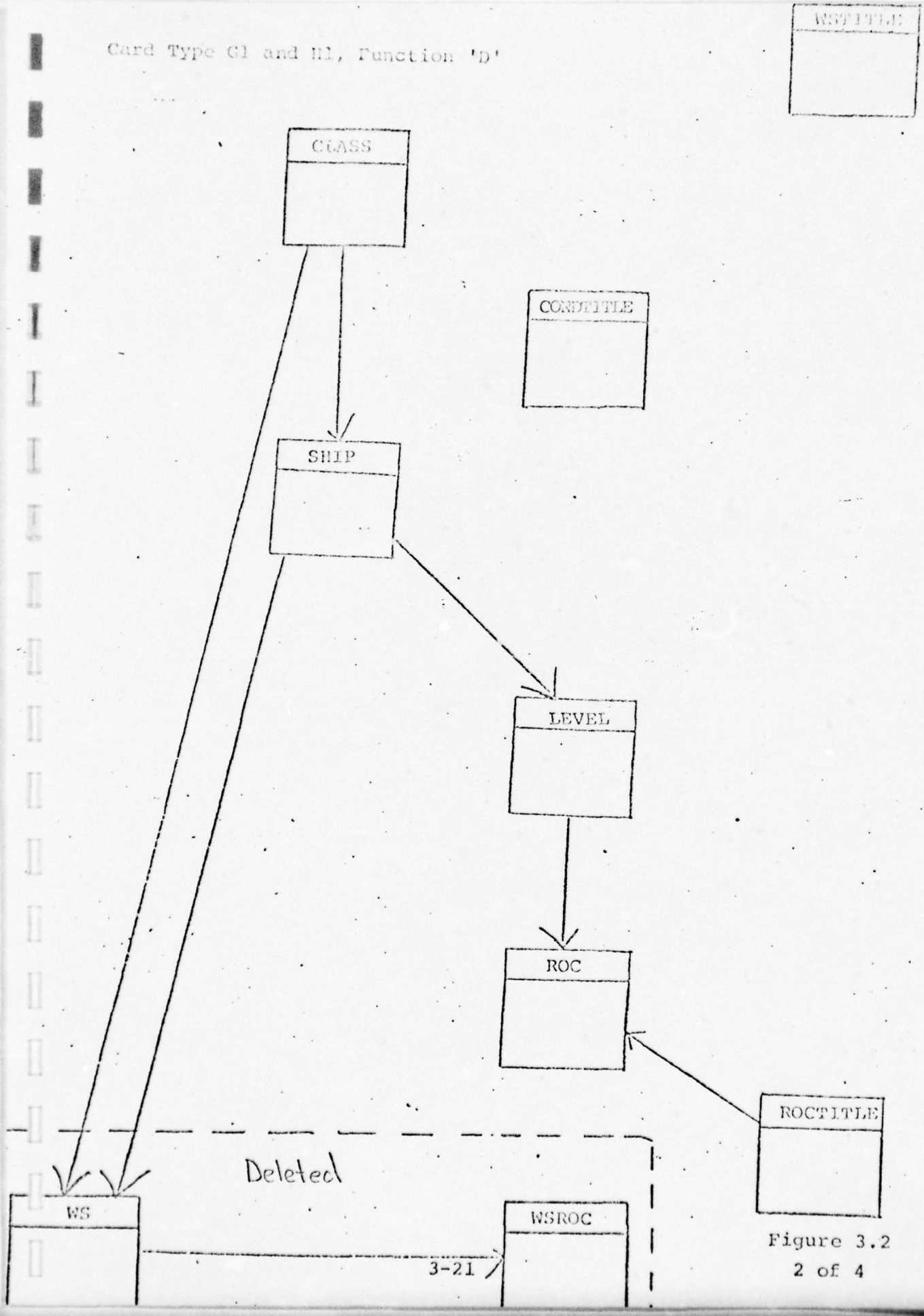


Figure 3.2
2 of 4

Card Types G5 and H5, Function 'D'

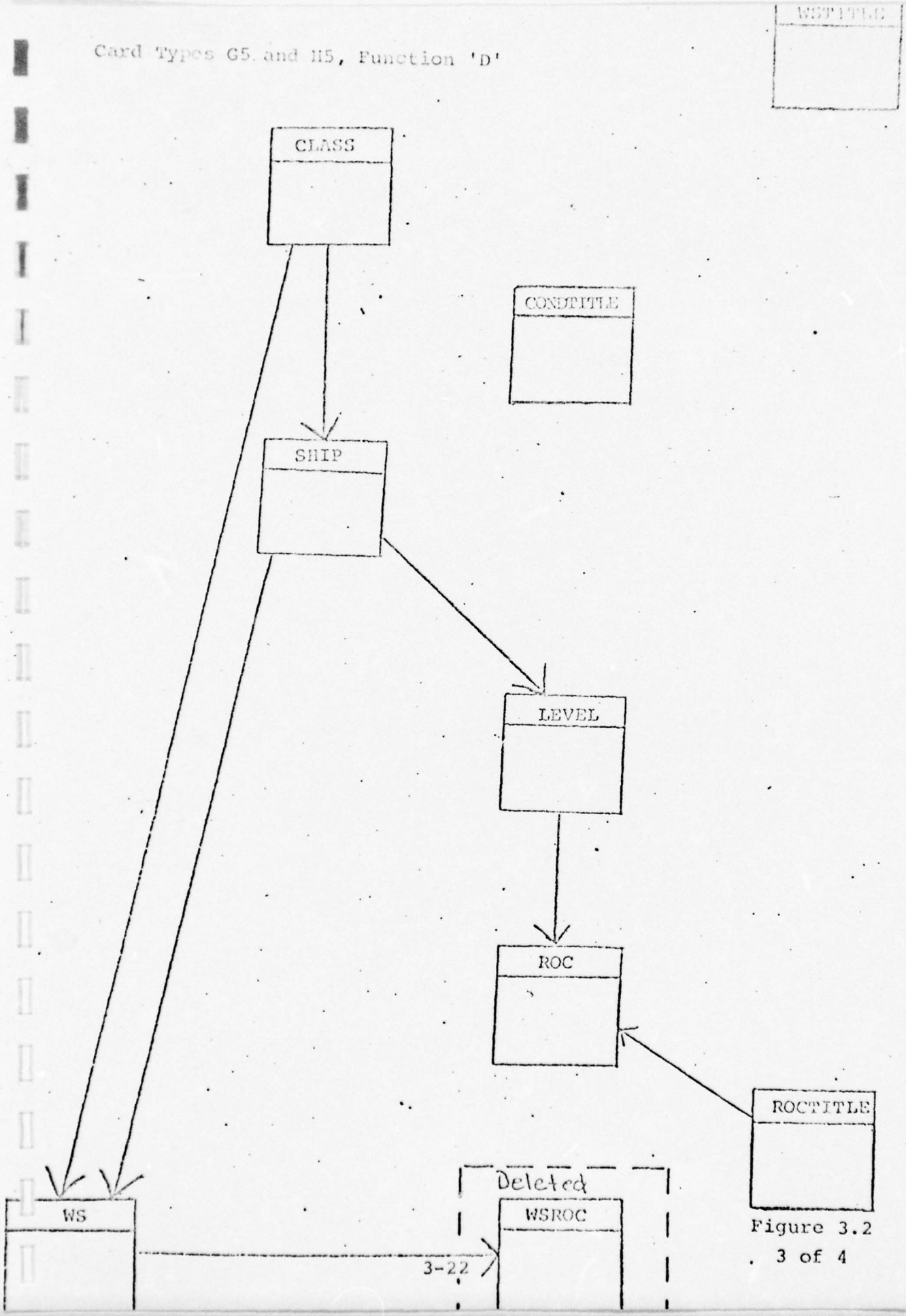


Figure 3.2
. 3 of 4

Card Type J1, Function 'D'

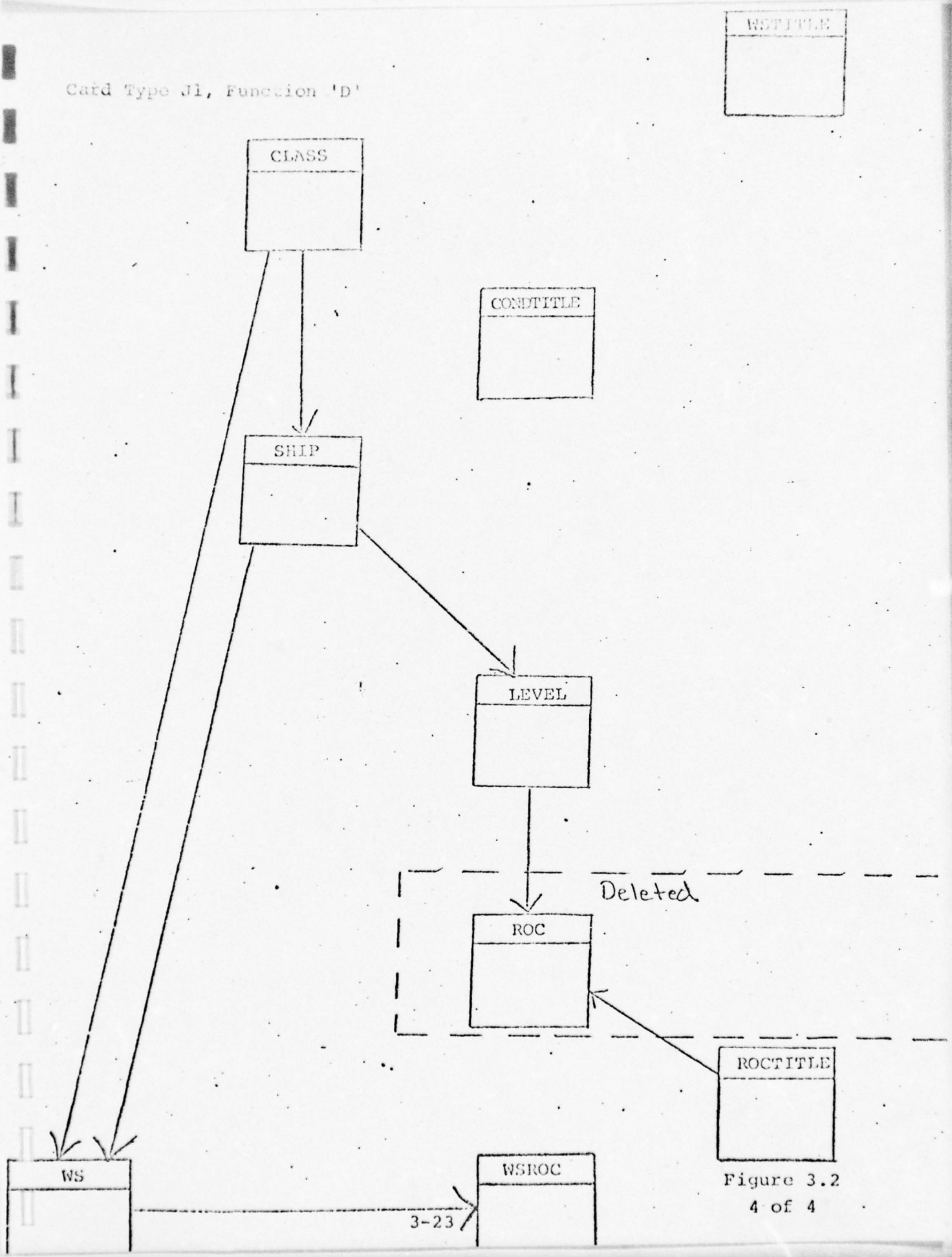
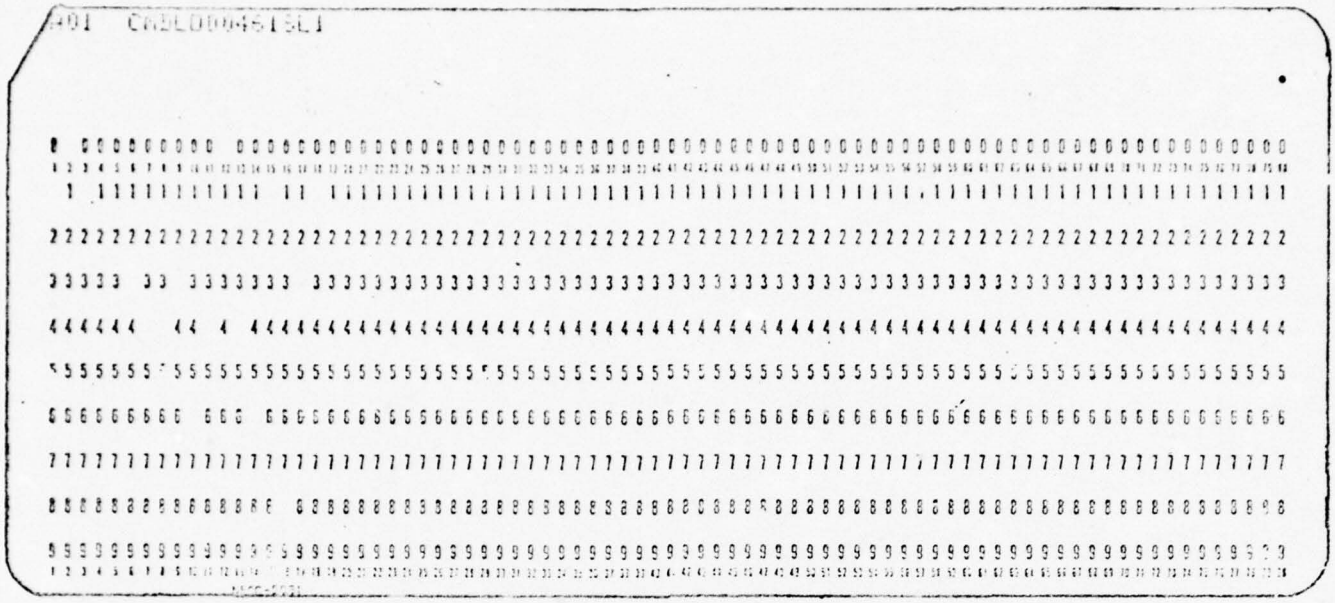


Figure 3.2
4 of 4

3.5.4 Watchstation Generation Subsystem



The sample displayed has a Transid of 'A', a sequence number of '01', Record ID 'ΔΔ' and Transcode 'CMD'. This card will produce a minimum list of watchstations required to be manned under each condition for ship '04618' and tasking level 'L1'.

3.6 Output Requirements

3.6.1 Load and Update Subsystems

The same types of output are generated by the Load, Update (1) and Update (2) Subsystems. These include error reports identifying card input containing invalid data, edit total reports identifying various functional totals by card type and record type, and displays of Database records that have been added or modified. The output will be produced every time one of the subsystems is run due to the need to update or load data on the Database. Output from Update Subsystem (2) may contain classified information, but handling of this information has not yet been defined. the output will be distributed to a Manpower Analyst who will be identified by User Interface Requirements to be defined for NMRS. Also considered as associated output are updated records on the Database and new records on the Database.

3.6.2 Watchstation Generation Subsystem

Output from the Watchstation Generation Subsystem will be produced randomly, as a function of the need to produce a ship manpower document for the first time or due to any changes to the ship ROC which might impact the watchstations. The output will be in the form of a disk file, which will be converted to ISAM organization to be input to the Document Input Processor (DIP).

3.7 Utilization of System Outputs

In general, there are two major NMRS objectives to which the ROC/Watchstation Module responds: Automated manpower document production and a flexible manpower management information system. The utilization of outputs of the ROC/Watchstation Module are discussed in the context of those objectives.

The principal system output is a file of minimum watchstation requirements (with skill categories, levels, and specialties identified) associated with a specified set of operational requirements called Required Operational Capabilities (ROC) which have been assigned to an individual ship. This output, provided to the NMRS Document Input Processor (DIP), defines the operational manpower workload in sufficient detail that it can be processed in Billet Derivation (BILDER) with maintenance-related workload to form billets. Billets are the basic manpower units of interest to the users of Ship Manpower Documents. In summary, the aggregate of minimum watchstation requirements output from the ROC/Watchstation Module form an essential input to the BILDER process - operational workload requirements.

Looking at NMRS as a flexible manpower management information system, other outputs from the ROC/Watchstation Module are needed. In this context, NMRS must provide information to manpower managers which assesses the manpower impact of changing the input criteria from the standards used for document production to some other set of conditions. For example, if a Ship Manpower Document defines the manpower requirements for a ship to meet all maintenance standards while existing at sea at war, a manpower manager may wish to examine the outcomes of varying that environment. The ROC/Watchstation Module must be able to respond to changes to input criteria where watchstation requirements are involved. The following are examples of this type of variability:

1. Reduce the provision for flight quarters from 16 hours to 12 hours per day.

2. Eliminate specific Sub-Operational Capabilities from the ship's ROC.

3. Increase a ship's underway replenishment requirements from 36 hours per week to 72.

4. Identify specific Condition III watchstations to be manned (when necessary) on a two-section basis with a correspondingly reduced Condition III endurance (i.e., from 60 days to 14 days).

In cases such as those described above, the ROC/Watchstation module will provide revised listings of minimum watchstation requirements for analysis purposes, or a revised input to DIP in order to permit an overall billet impact assessment.

3.8 System Query Capabilities

The System Query Capabilities for the Ship/ROC Watchstation Module Database exist in the form of IDMS/CULPRIT Reports. The reports that may be generated using catalogued procedures are the following: (1) Level/ROC Report, (2) Ship/ROC/Watchstation Report, (3) Condition Titles Report, (4) ROC Titles Report, and (5) Watchstation Titles Report. The Level/ROC report prints the contents of the Level and ROC files on the Database indicating the relationships between the records on the files. The Ship/ROC/Watchstation Report may be printed in two possible ways: (1) The user may print the contents of the Ship, WS, and WSROC files. This report is formatted in a manner to clearly show the interrelationships among these record types. (2) The user may generate the report for one ship only. The report in this form will identify all of one ship's watchstations and their associated sub-operational capabilities. In order to obtain the report in this form, two cards must be added to the run stream that would print the report in its generalized format. The Users Manual shows card formats. The Condition Titles Report prints the contents of the CONDTITLE file, displaying all conditions and their associated condition names. The ROC Titles Report prints the contents of the ROCTITLE file, identifying sub-operational capabilities with their four digit ROC numbers. The Watchstation Titles Report prints the contents of the WSTITLE file, identifying watchstation numbers and their associated titles.

Below are listed the reports that are being generated:

1. ROC DICTIONARY
2. CONDITION TITLES
3. ROC TITLES
4. WATCHSTATION TITLES
5. SHIP/ROC/WATCHSTATION REPORT

3.9 Query Preparation

Since all of the reports generated are run using catalogued procedures, the only user request will be that a certain report be produced. The documentation package, Culprit Reports of Ship/ROC/Watchstation Database identifies and displays the catalogued procedures used to produce each report. The user will specify which form of the Ship/ROC/Watchstation report he wants and will provide the appropriate input cards when necessary.

3.10 Control Instructions

Since the Cullinane IDMS/CULPRIT Manual Release Number 3 explains the technical aspects of CULPRIT and its procedure for generating reports when used against an IDMS data base, there exists no need for further elaboration here. The IDMS/CULPRIT routine functions as an output processor in that it allows the user to generate a variation of reports and/or files without changing or affecting the data base.

The user manual gives examples of the JCL necessary to generate reports using CULPRIT. These examples, along with the file descriptions and schema definitions of the SHIP/ROC/WATCHSTATION MODULE data base, were used as guidelines for creating the necessary JCL.

APPENDIX A

MODULE NMWV01

Module Description

NMWV01 validates the card type for all input transactions to module NMWV03. The module sorts the valid transactions by card type and outputs the different types to individual files. The program was written under contract number N-00014-76-C-0473, effective 24 Nov 75.

Module Functions

The following functions are performed by NMWV01:

A. Edit all input transactions for card type 'A1', 'B1', 'C1', 'D1', or 'F1'.

B. Sort valid transactions by card type and write the different card types to individual output files.

C. Print an error report of those transactions with invalid card type.

D. Print totals of records read, records written, and records with errors.

Inputs

Input to NMWV01 consists of 5 types of card input transactions. The 5 types are identified in card columns one and two. Valid types are 'A1', 'B1', 'C1', 'D1', and 'F1'. Cards with invalid card types will be accepted by the program and printed on an edit report, but will not be output for processing by the Load program. The different input types are described in the Users Manual. The Input File and Transaction formats may be seen in the Module Specifications.

Outputs

Output from NMWV01 will consist of up to five temporary files and two reports. A file will be output for each different type of valid input transaction. These files will be input to individual utility sorts and will then be passed to the Load program. There will always be a report identifying the number of input card transactions read, the number written to the output files, and the number having invalid card types. An edit report identifying transactions with invalid card types will be printed when they do occur. The different output files are described in the Users Manual; and output file formats and printer layouts may be seen in the Module Specifications.

Working Storage

Working Storage consists of report headers and detail line formats, a table used for printing the current date on the edit report, counters used to accumulate totals, and subscripts used to access table data. The different data elements in the Working Storage Section are described in the Remarks paragraph of the Cobol Source Listing.

Program Logic

The first step in the processing is to open files and initialize print areas.

When a card is read, an accumulator for transactions read is incremented. The card is edited for a valid card type. If the card type is valid, a routine is performed that writes the transaction to an output file based on the card type, then there is a branch to read another card. If the card has an invalid card type, it is printed on an edit report, an accumulator for transactions with errors is incremented and there is a branch to read another card.

When processing of the input file is complete, edit totals are printed, files are closed, and program processing is ended.

PROGRAM FLOWCHART
NSAVO1

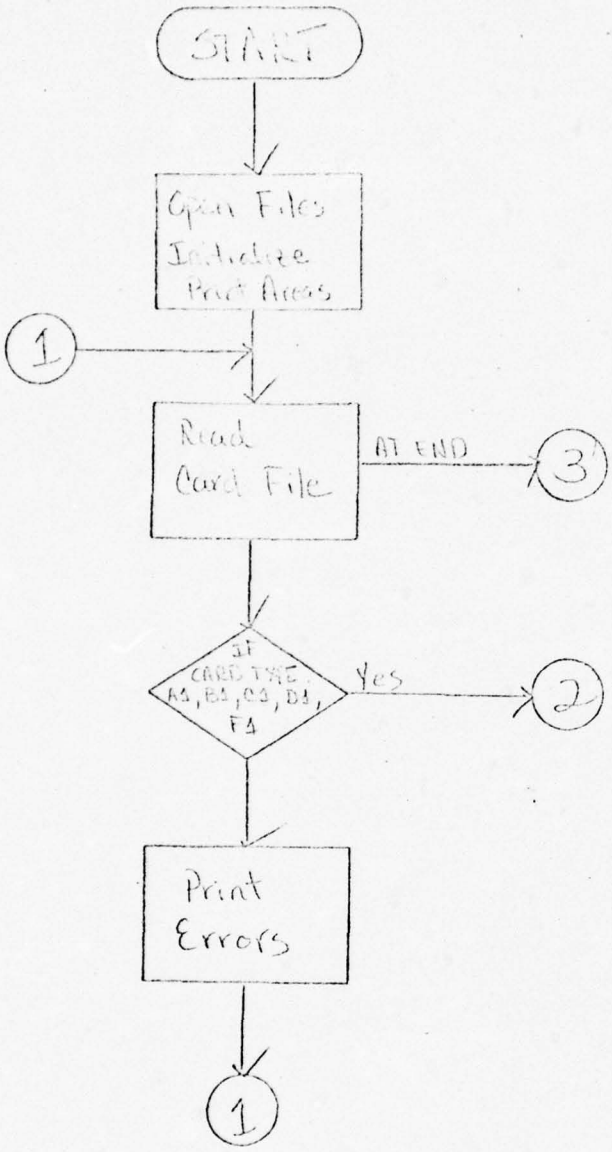
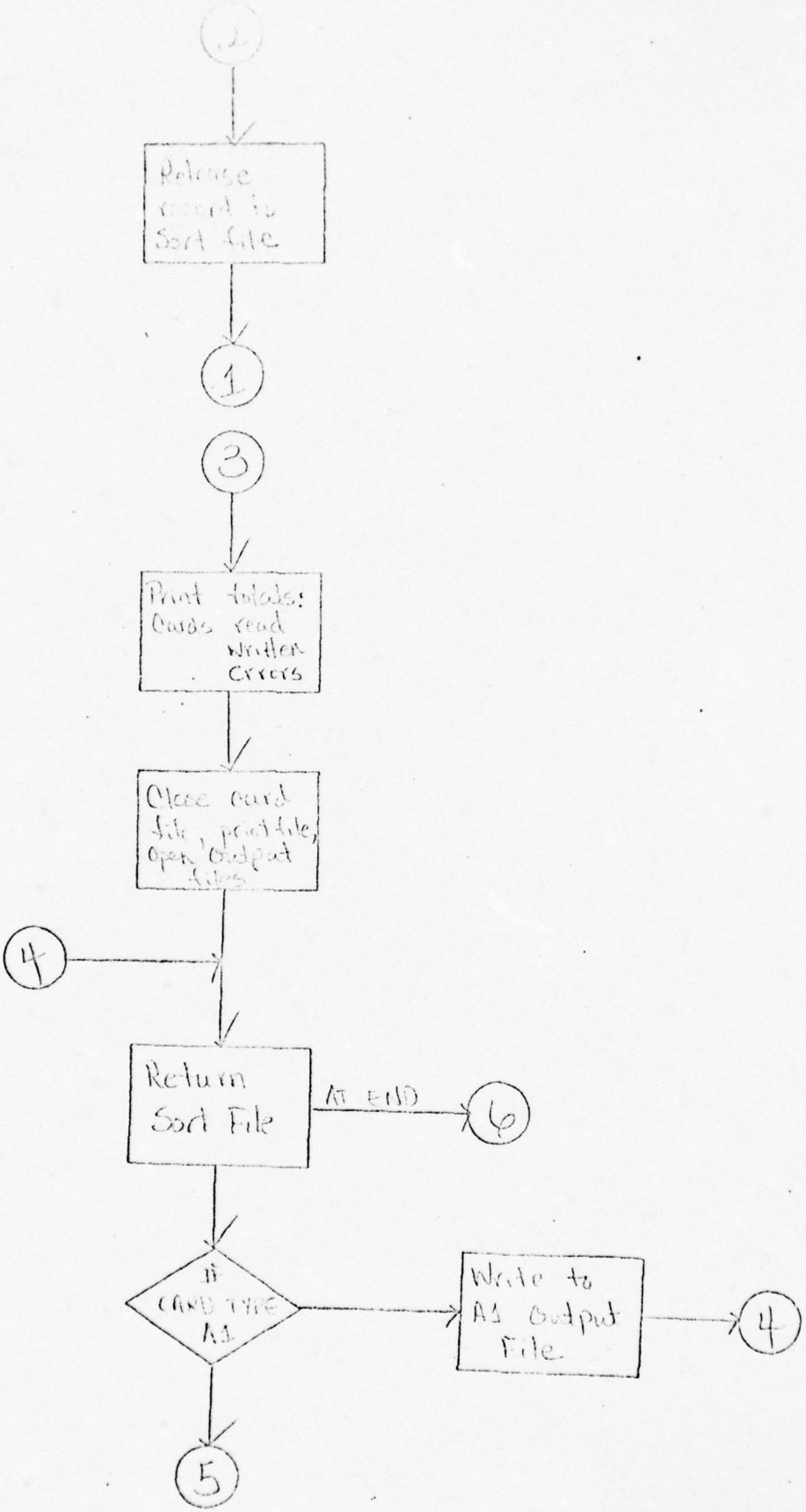
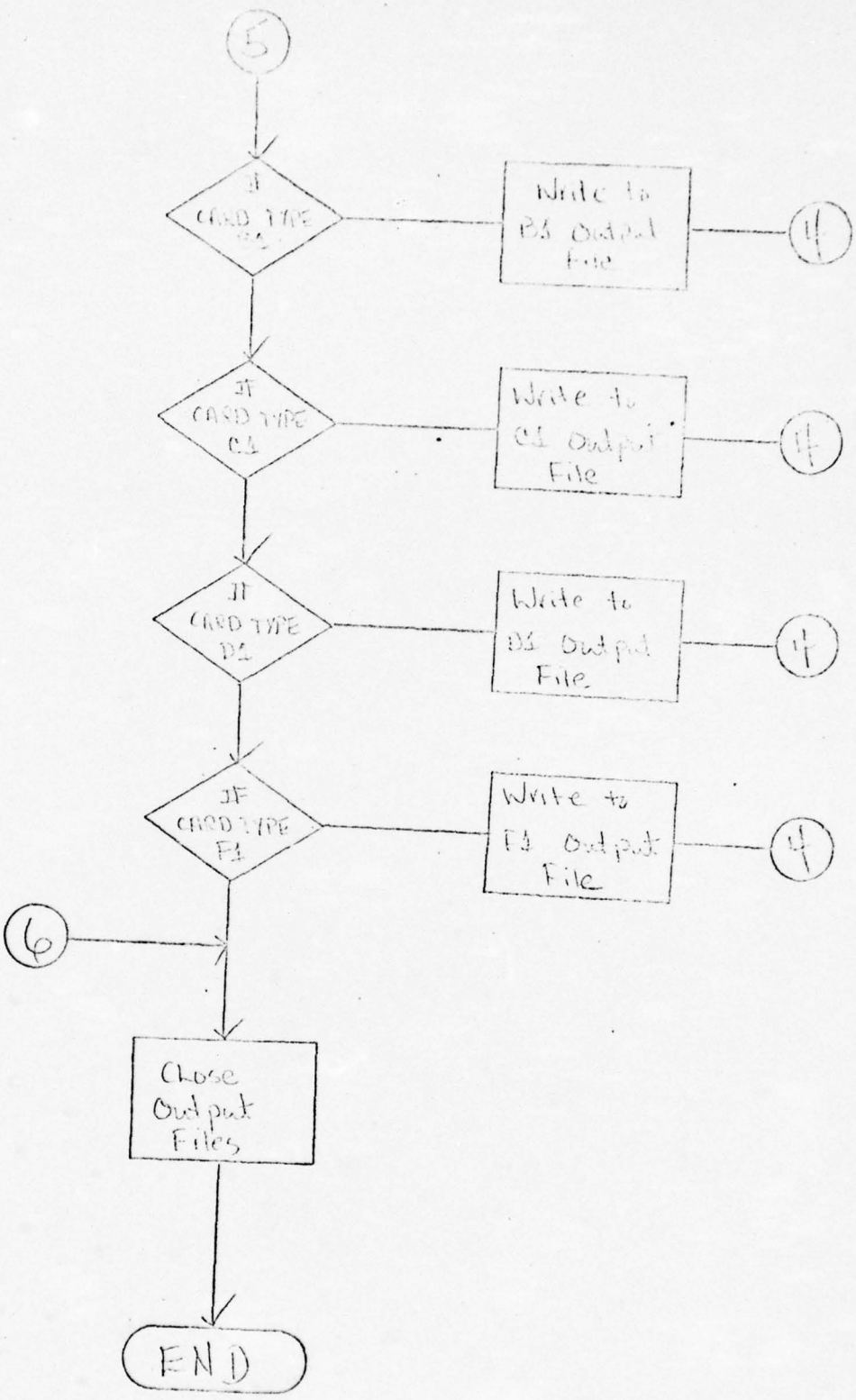


Figure A-1


2
Release record to Sort file
1
3
Print totals: Cards read Writen Errors
Close card file, print file, Open Output files
4
Return Sort File
AT END
6
If CARD TYPE A1
Write to A1 Output File
4
5


5
IF CARD TYPE B1
Write to B1 Output File
4
IF CARD TYPE C1
Write to C1 Output File
4
IF CARD TYPE D1
Write to D1 Output File
4
IF CARD TYPE F1
Write to F1 Output File
4
6
Close Output Files
END

APPENDIX B

MODULE NMWV02

Module Description

NMWV02 validates the card type and transcode for all input transactions to modules NMWB01 and NMWB02. The module sorts the valid transactions by card type and outputs the different types to individual files. The program was written under contract number N-00014-76-C-0473, effective 24 Nov 75.

Module Functions

The following functions are performed by NMWV02:

A. Edit all input transactions for transcode 'ROC' and card type 'A1', 'B1', 'C1', 'C2', 'D1', 'F1', 'G1', 'H1', 'G5', 'H5', or 'J1'.

B. Print an error report of those transactions with invalid transcode and/or card type.

C. Print totals of records read, records written, and records with errors.

D. Sort valid transactions by card type and write different card types to individual output files.

Inputs

Input to NMWV02 consists or 10 types of card input transactions identified in card columns 10-11. The valid types are 'A1', 'B1' 'C1', 'D1', 'F1', 'G1', 'H1', 'G5', 'H5', and 'J1'. Cards with invalid card types will be accepted by the program and printed on an edit report, but will not be output for further processing.

Outputs

Output from NMWV02 will consist of up to four temporary files and two reports. Which files are output will depend on the validity of the input transactions and to which update program they will eventually be input. Basically one file for each valid type of transaction will be written. There will always be a report identifying the number of input card transactions read, the number written to the output files, and the number having invalid card types. An edit report identifying transactions with invalid card types or invalid NMRS Transcodes will be printed when these errors occur.

Working Storage

Working Storage consists of report headers and detail line formats, a table used for printing the current date on the edit report, counters used to accumulate totals, and subscripts used to access table data. The different data elements in the Working Storage Section are described in the Remarks paragraph of the Cobol Source Listing

Program Logic

The first step in the processing is to open files and initialize print areas.

When a card is read, an accumulator for transactions read is incremented. The card is edited for a valid card type and NMRS Transcode. If these two fields are valid, a routine is performed that writes the transaction to an output file based on the card type, an accumulator for transactions written is incremented, and there is a branch to read another card. If the card has invalid data, it is printed on an edit report, an accumulator for transactions with errors is incremented and there is a branch to read another card.

When processing of the input file is complete, edit totals are printed, files are closed, and program processing is ended.

PROGRAM FLOWCHART
NMNV02

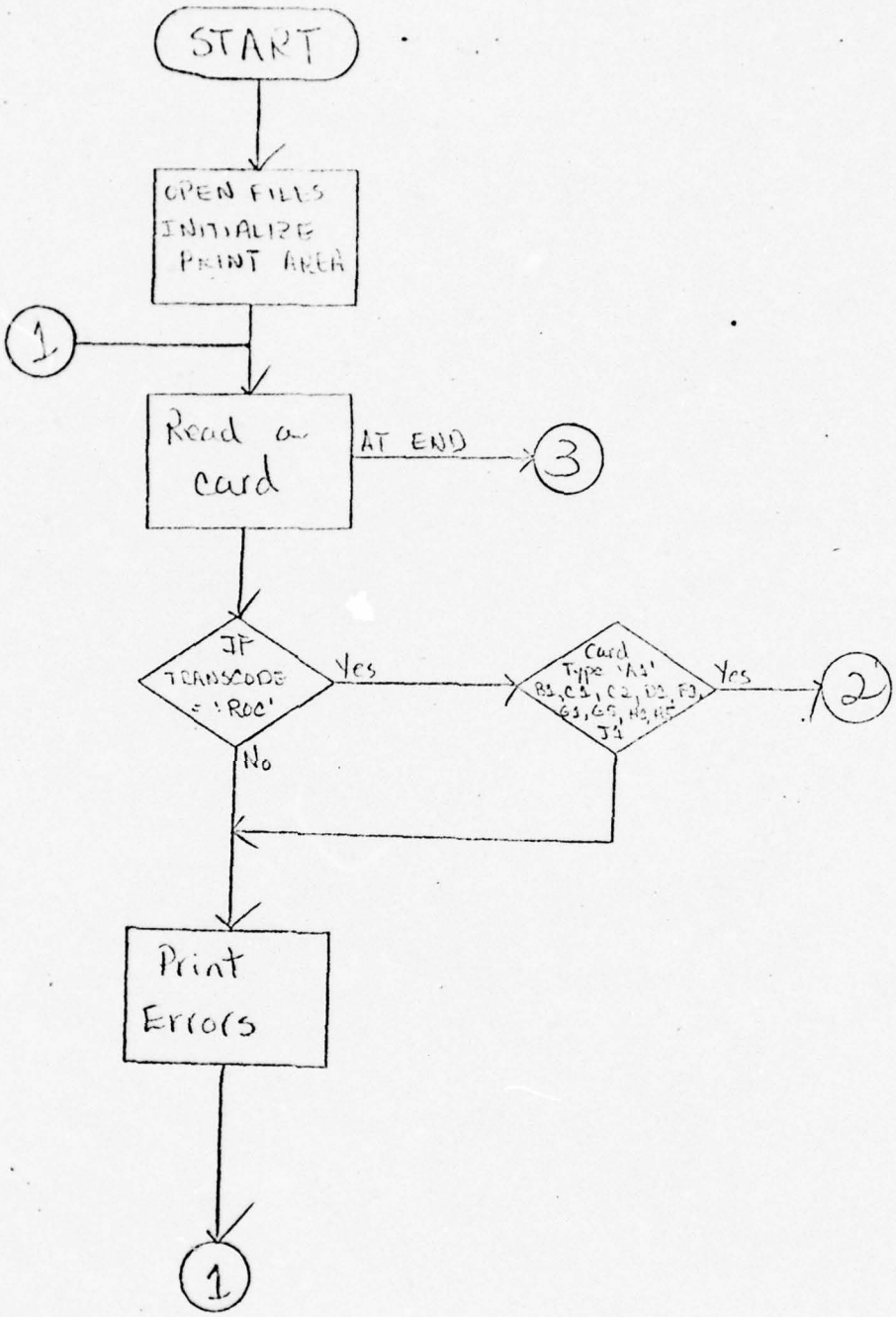
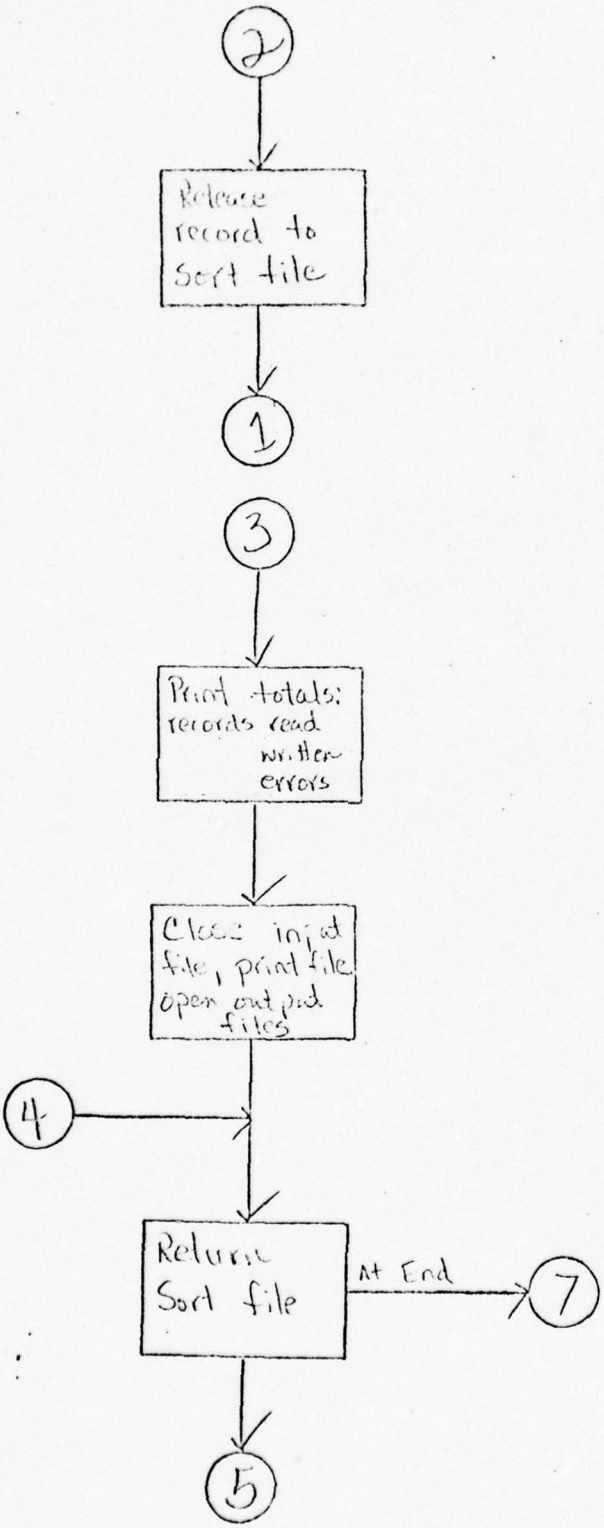
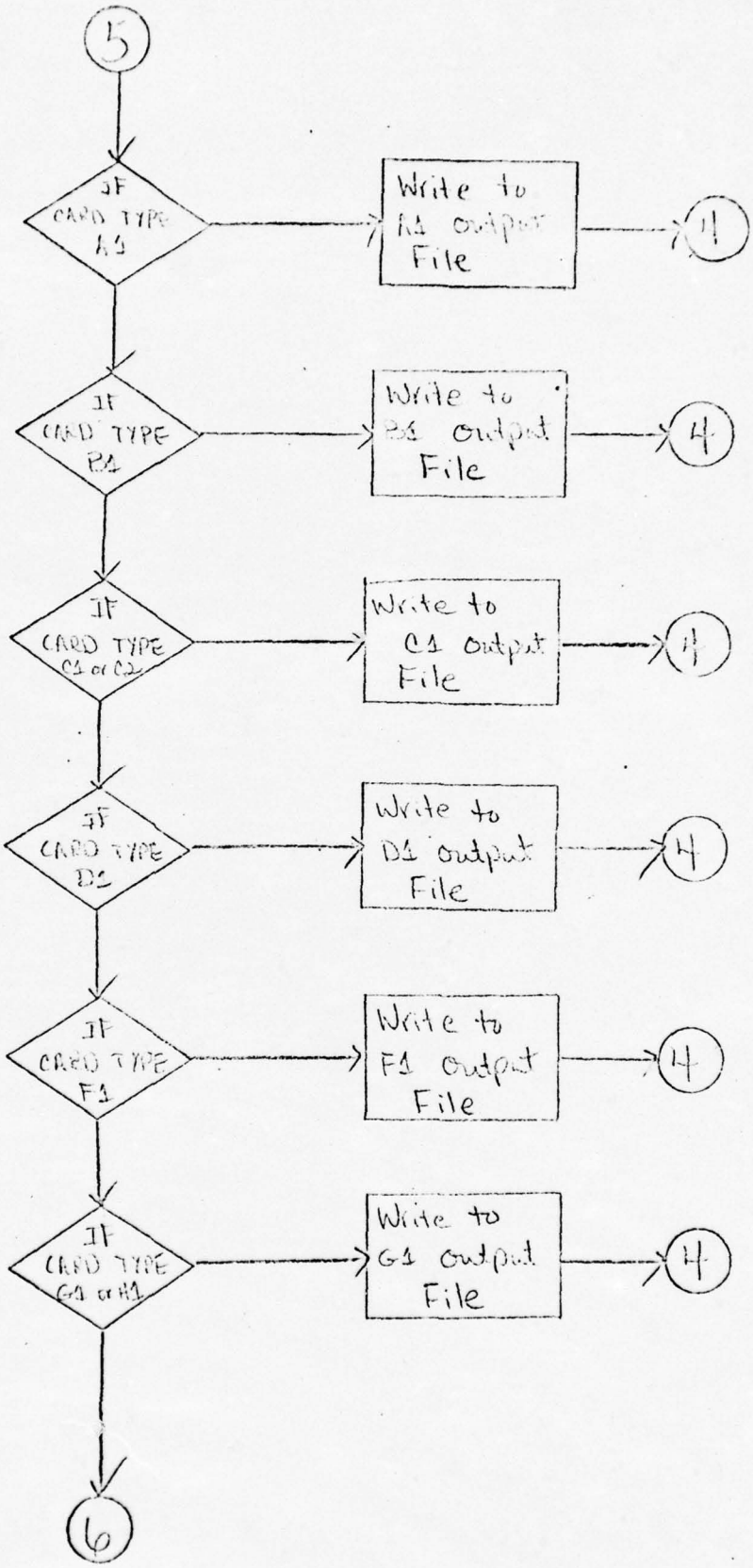
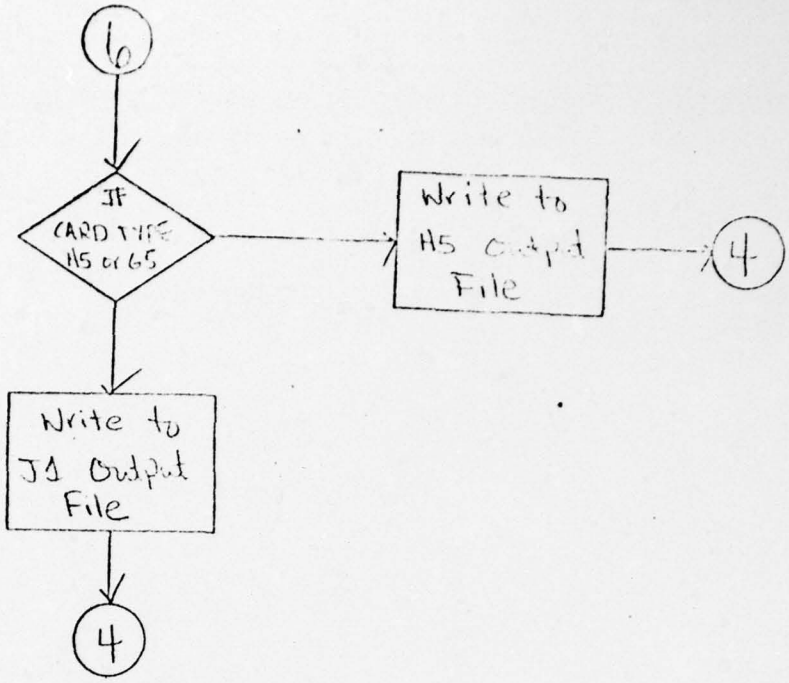
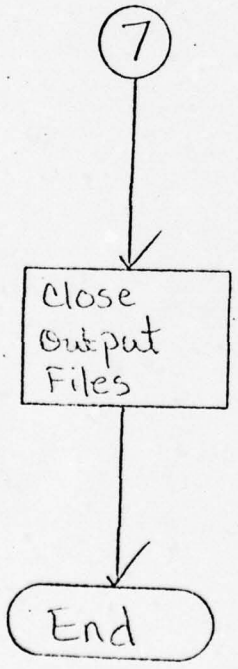


Figure B-1


2
Release record to Sort file
1
3
Print totals: records read written errors
Close input file, print file, open output files
4
Return Sort file
At End
7
5


5
IF CARD TYPE A1
Write to A1 output File
4
IF CARD TYPE B1
Write to B1 output File
4
IF CARD TYPE C1 or C2
Write to C1 output File
4
IF CARD TYPE D1
Write to D1 output File
4
IF CARD TYPE F1
Write to F1 output File
4
IF CARD TYPE G1 or H1
Write to G1 output File
4
6


6
IF
CARD TYPE
H5 or G5
Write to
H5 Output
File
4
Write to
J1 Output
File
4


7
Close
Output
Files
End

APPENDIX C

MODULE NMWV03

Module Description

This module loads the following record types onto the SHIP ROC/Watchstation Module Database: WSTITLE, ROCTITLE, SHIP, CLASS, LEVEL, and CONDTITLE. The program was written under contract number N-00014-76-C-0473, effective 24 Nov 75.

Characteristics

a. IT-REC is the card input transaction. The file is a concatenation of up to five individual files that are output from utility sorts.

b. Each input transaction is edited according to specifications for the particular record type associated with the transaction. These requirements are stated in '2.2 Module Functions'. If the record contains valid data, the transaction is used to load a particular Database record; otherwise the transaction is printed on an error report where the invalid card data is identified.

c. The data in positions 1-2 is the card type which identifies which Database record is to be added. The rest of the card data is used to build the Database records.

d. All of the transaction data is static.

e. The input transaction file is a temporary disc file that requires one 3330 disc track per 156 input records.

f. The sequence within the individual files is determined by utility sorts and will be listed below.
   A1: Ascending on W-A1-WATCHSTATION
   B1: Ascending on W-B1-ROCNO
   C1: Descending on W-C1-IDENT, Ascending on W-C1-CLASS, W-C1-UIC
   D1: Ascending on W-D1-CONDITION
   F1: Ascending on W-F1-SHIPID, W-F1-LEVELID

Data Environment

The purpose of W-MONTH-TABLE is to provide abbreviations for the twelve months in order to include the alpha month on report headings. The table values are hard-coded in the program. The subscript used to access the table is W-MM-IND. W-ERROR-MESSAGES holds comments that are printed on an error report to identify incorrect data on the input transactions. The subscript used to access the table is W-CRD-ERRS.

Program Logic

Note: Throughout the program whenever the Database is accessed, the error status field is examined. When the value of the error status indicates a problem, an error routine is performed, edit totals are printed, and processing is ended. This logic will not be included in the description below.

Program processing begins with all files being opened, and database areas being made ready. Headers and print areas are initialized, and then file processing begins.

When an input transaction (will refer to input transaction as 'card') is read, all of the card data except card type is moved into Working Storage. The card type is then examined, and a branch is made to appropriate edit routine.

If the card type is 'A1', the processing is as follows: The Watchstation number is compared to the last one read. If the two are equal, indicating duplicates, an error message is stored. If the Major Functional Area is not numeric, an error message is stored. If Minor Functional Area is not numeric and equals spaces, the spaces are replaced with zeros. If it is not numeric and does not equal spaces, an error message is stored. If the sequence code equals spaces, they are replaced with zeros. If it is not numeric, an error message is stored. If the Minor Functional Area equals zeros and Sequence code does not equal zeros, an error message is stored. If the Watchstation title is not present, an error message is stored. An accumulator for 'A1' cards read is incremented. If there were no card errors a routine to load the database is performed (this will be described later), and the Watchstation number is stored. If card errors were found, a routine to print error messages is performed (described later). The program now branches to read another card.

If the card type is 'B1', the processing is as follows: the ROC number is compared to the last one read. If the two are equal, an error message is stored. If the ROC number is not numeric, an error message is stored. If the Mission Area is blank, an error message is stored. If the Mission Area is not alphabetic, an error message is stored. If the Operational Capability Code is missing, an error message is stored and there is a branch to a routine to edit the Required Functional Capability (RFC) Code. If the Operational Capability Code is numeric, but not right justified, an error message is stored and there is a branch to a routine to edit RFC code. If the Operational Capability code is right justified, but not numeric, an error message is stored. If the RFC is numeric or equals spaces there is a branch to an end of edit routine. If RFC code is numeric but not right justified, an error message is stored, and there is a branch to an end of edit routine. If RFC code is left justified and not numeric, an error message is stored and there is a branch to an end of edit routine. If the RFC code is right justified and numeric there is a branch to an end of edit

routine, otherwise an error message is stored. An accumulator for 'B1' cards read is incremented. If there were no card errors, a routine to load the database is performed and the ROC number is stored. If card errors were found, a routine to print error messages is performed. The program now branches to read another card.

If the card type is 'C1' the processing is as follows: If the record identifier is 'CLS' and the class is equal to the class in hold, an error message is stored. If the record identifier is blank and the ship is equal to the ship ID in hold, an error message is stored. If class is not numeric an error message is stored. If the ship ID is blank, an error message is stored. If the record identifier is not equal to either 'CLS' or spaces, an error message is stored. If shiphull is blank, an error message is stored, and if ship name is blank, an error message is stored. An accumulator for 'C1' cards read is incremented. If there were no card errors, a routine to load the database is performed and the ship number is stored. If card errors were found, a routine to print error messages is performed. The program now branches to read another card.

If the card type is 'D1', the processing is as follows: The condition is compared to the condition in hold. If the two are equal, an error message is stored. If the condition title is blank, an error message is stored. An accumulator for 'D1' cards read is incremented. If there were no card errors, a routine to load the database is performed and the condition is stored. If card errors were found, a routine to print error messages is performed. The program now branches to read another card.

If the card type is 'F1', the processing is as follows: The level and ship code are compared to the level and ship values that have been stored. If they are both equal to the respective values in hold, an error message is stored. If the level is blank, an error message is stored. If the level name is blank, an error message is stored. If the ship code is not numeric, an error message is stored. An accumulator for 'F1' records read is incremented. If no card errors were detected, a routine to load the database is performed; ship code and level are stored. If card errors were found, a routine to print error messages is performed. The program now branches to read another card.

The following is the logic for the routine to print error messages: If the line count exceeds 43, a routine to begin a new page is performed. The input transaction is moved to the print line and is printed. The card is underlined and another line is printed with spaces. The line count is incremented. There is a loop to print the error messages: A subscript (W-ERR-IND) is incremented by one. If the subscript is greater than the number of errors that occurred (W-CRD-ERRS), the loop is ended. There is a routine performed to determine whether to go to a new page and print headings if necessary. An error message is moved

from a table (W-PRT-ERR) to print line and is printed. The line count is incremented. Then there is a branch to the beginning of the loop which continues until all error messages are printed. Once out of the loop, an accumulator for card type errors is incremented, two lines of spaces are printed, the table that held messages and the area that held the card in error are cleared, and the subscript and the error counter are re-initialized to zero.

The header routine sets the line count back to zero and adds 1 to the page counter. It then prints headings on a new page by moving to the print line headers that are set up in Working Storage and printing them one by one.

The line check routine checks to see if the line count is greater than 46, and if it is, the header routine is performed.

The routine to print a line simply writes the line and then re-initializes the print line to spaces.

The following paragraphs will describe the routines to load data onto the database. The card types are checked, and a branch is made to the appropriate load routine, or the program falls into the routine to load a WSTITLE record when the card type is 'A1' which follows: Data is moved from the card to the appropriate WSTITLE fields in Working Storage, and then the record is stored. If the record is a duplication of a record that already exists, an error message is stored and the program branches to an exit routine. The record that was added is displayed, then there is a branch to an exit routine.

If the record type is 'B1' the following process loads a ROCTITLE record. Data is moved from the card to the appropriate ROCTITLE fields in Working Storage, then the record is stored. If the error status indicates that this record is a duplication of one that already exists, an error message is stored and the program branches to an exit routine. If the record is added successfully, it is displayed and there is a branch to an exit routine.

If the card type is 'C1', either a CLASS or SHIP record may be loaded. The processing is as follows: If the record identifier is 'CLS', there is a branch to the 'CLASS' load routine. The class is used to obtain the CLASS record owner of the CLASS-SHIP set to which the SHIP will be added. If the CLASS record is not found, an error message is stored and there is a branch to an exit routine. If the CLASS record is found successfully, the ship data is moved from the card into the appropriate SHIP record fields in Working Storage, and the new SHIP record is stored. The record is automatically linked to its CLASS owner. If the error status indicates a duplicate, an error message is stored, and there is a branch to an exit routine. If the record is added successfully, it is displayed and there is a branch to an exit routine.

The routine to load CLASS record is as follows: The class data is moved from the card into the appropriate CLASS record fields in Working Storage and the new CLASS record is stored. If the error status indicates a duplicate, an error message is stored and there is a branch to an exit routine. If the record is added successfully, it is displayed and then there is a branch to an exit routine.

If the card type is 'D1' a CONDTITLE record is to be added and the processing is as follows: The card data is moved into the CONDTITLE record fields in Working Storage and the record is stored. If the error status indicates a duplicate, an error message is stored and there is a branch to an exit routine. If the record is added successfully, it is displayed and there is a branch to an exit routine.

If the card type is 'F1', a LEVEL record is to be added and the processing is as follows: The ship on the card is used to obtain the SHIP record owner of the SHIP-LEVEL set to which the LEVEL record is to be added. If the SHIP record is not found, an error message is stored, and there is a branch to an exit routine. If the SHIP is located successfully, the level data is moved from the card into the LEVEL record fields in Working Storage and the record is stored. If the error status indicates a duplicate, an error message is stored and there is a branch to an exit routine. If the record is added successfully, it is displayed and there is a branch to an exit routine.

The database error routine is performed whenever an error status that indicates a problem is returned. The card being processed is displayed, along with the error status and an indicator that locates the paragraph where the problem occurred. It branches to the routine to print edit totals.

The update exit routine checks to see if any errors occurred during the load routine. If there were none, the database key of the record just added is displayed, an accumulator for transactions added is incremented for the appropriate card type, and the program falls into the routine to print edit totals. If errors did occur, the routine to print error messages is performed, and the program falls into the routine to print edit totals.

The routine to print edit totals loops through a table where the totals are stored (W-CTR1-TABLE). Before the loop begins, headers are printed on a new page. The loop is as follows: A subscript (W-TYPE-IND) to identify the card type is incremented. If the subscript is greater than 5 (only five card types), the subscript is re-initialized to zero and there is a branch to an end of program routine. Another subscript (W-CTR-IND) is incremented (for the totals). If the subscript is greater than 3, it is re-initialized to zero and there is a branch to print the line. The data from the table is moved to the print line and printed. Then there is a branch to the beginning of the loop. The loop continues until all the edit totals are printed.

The last paragraph closes all files, closes the database areas, and ends program processing.

PROGRAM FLOWCHART
NMWV03

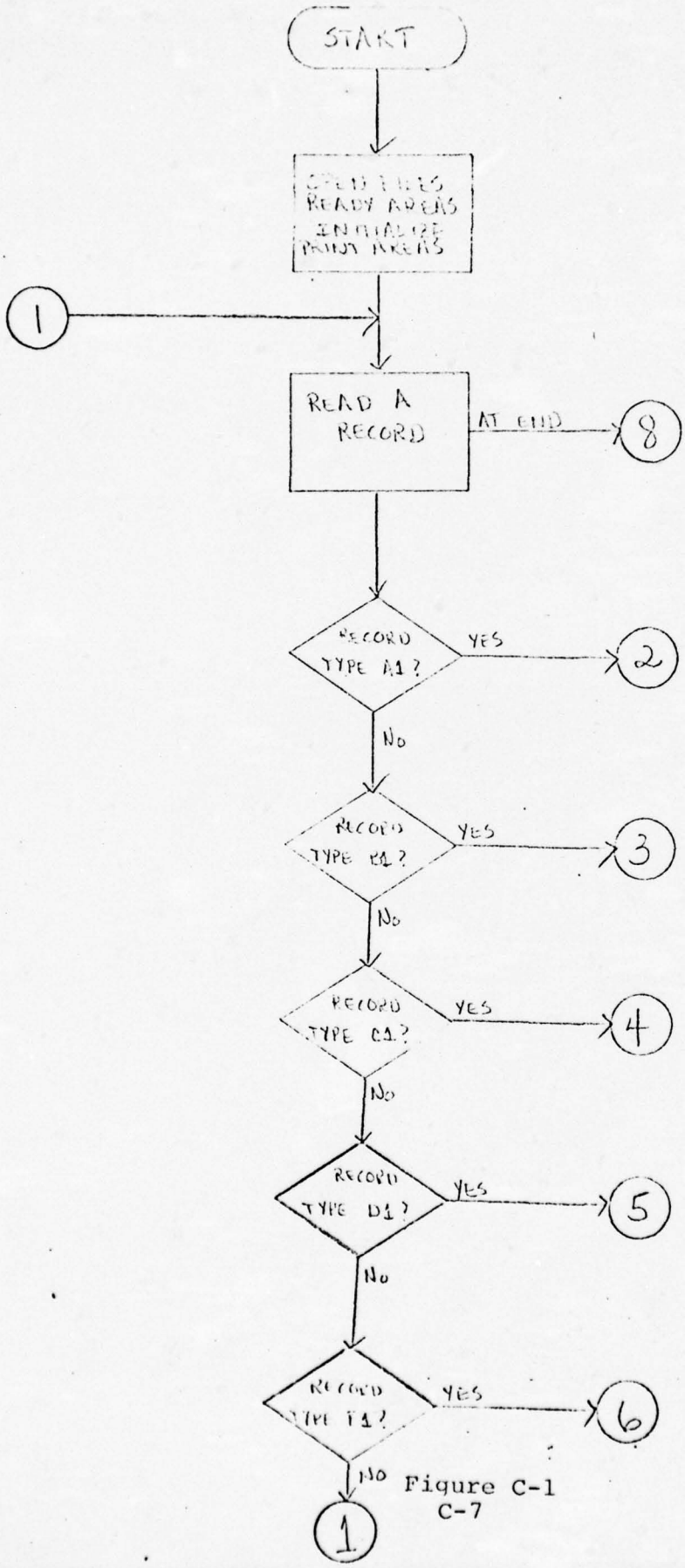
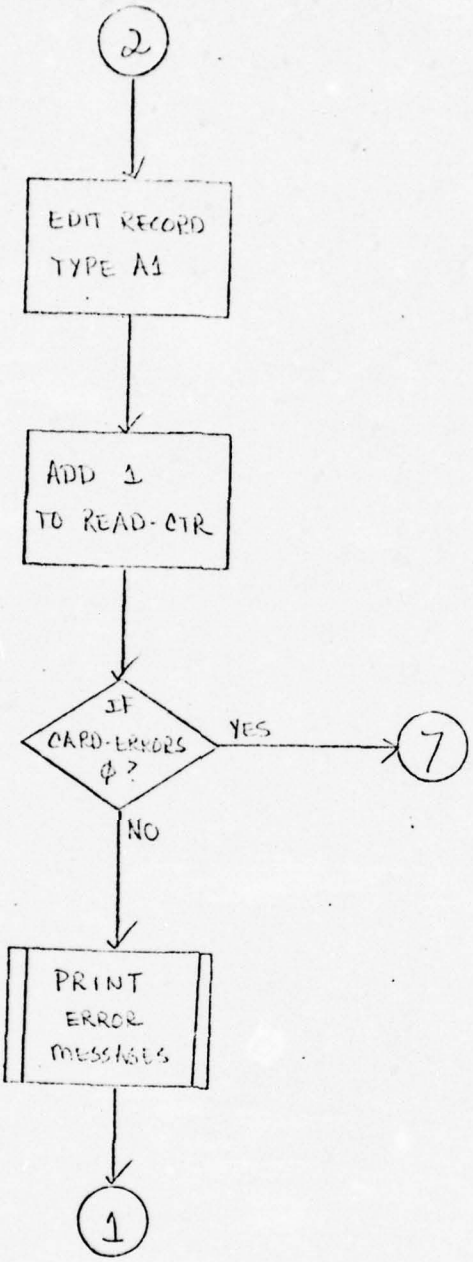
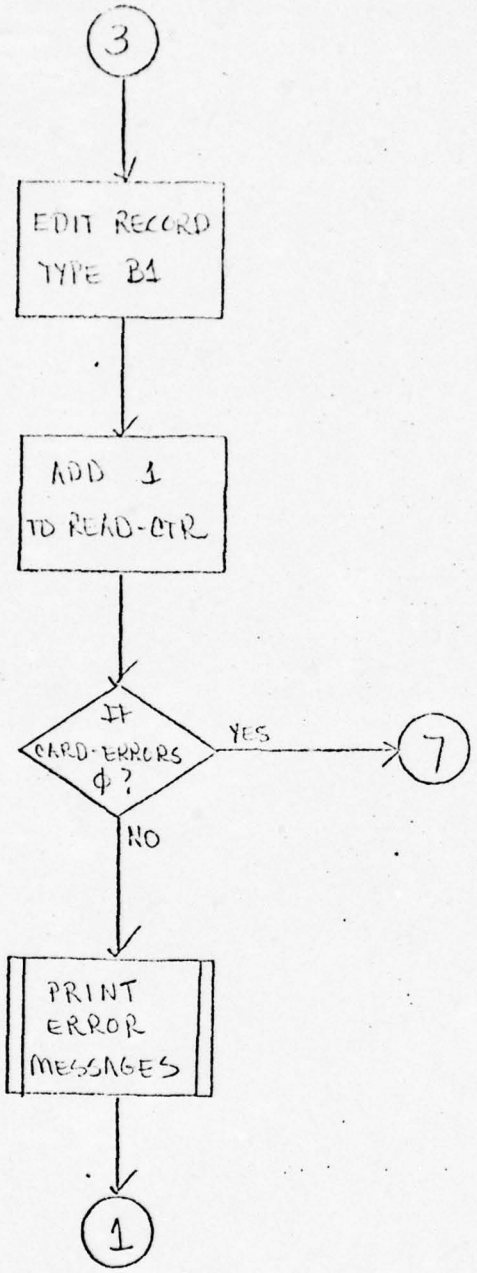
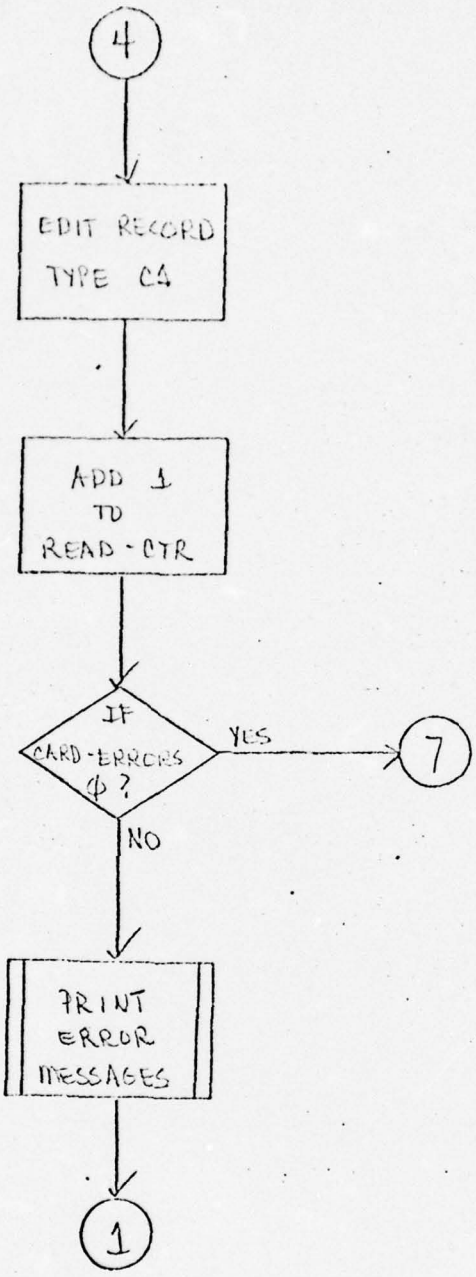
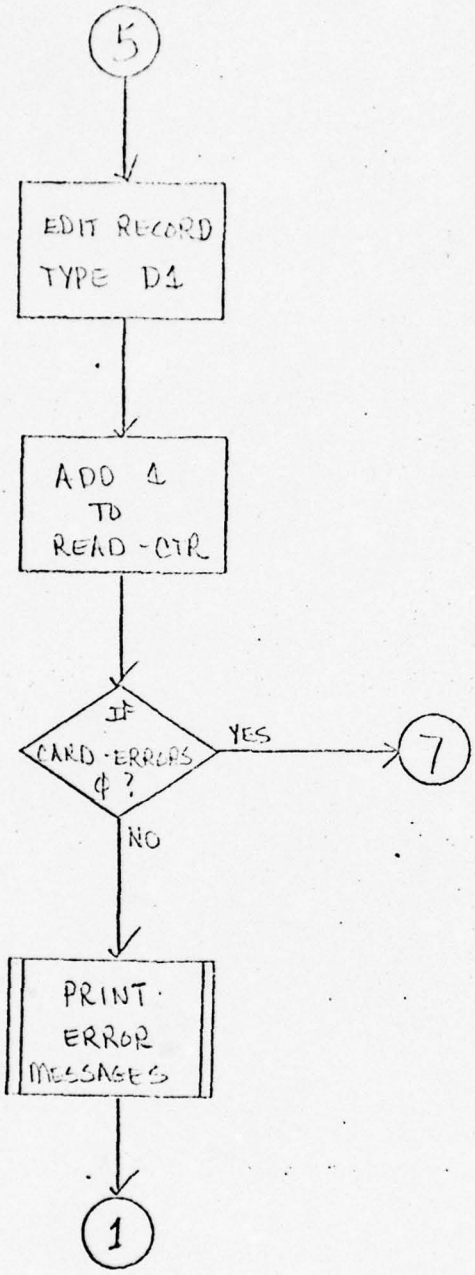


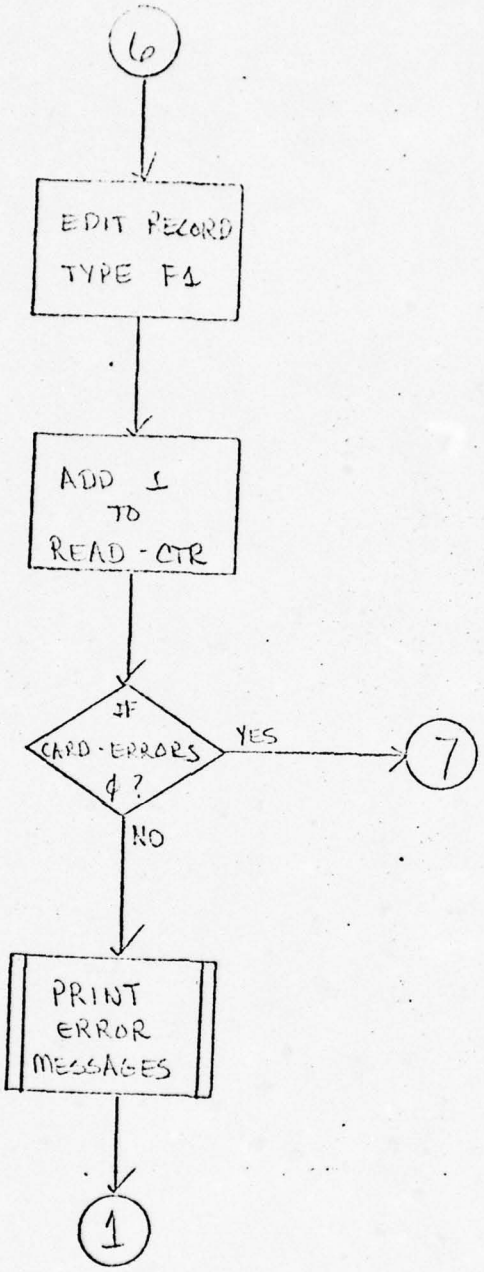
Figure C-1

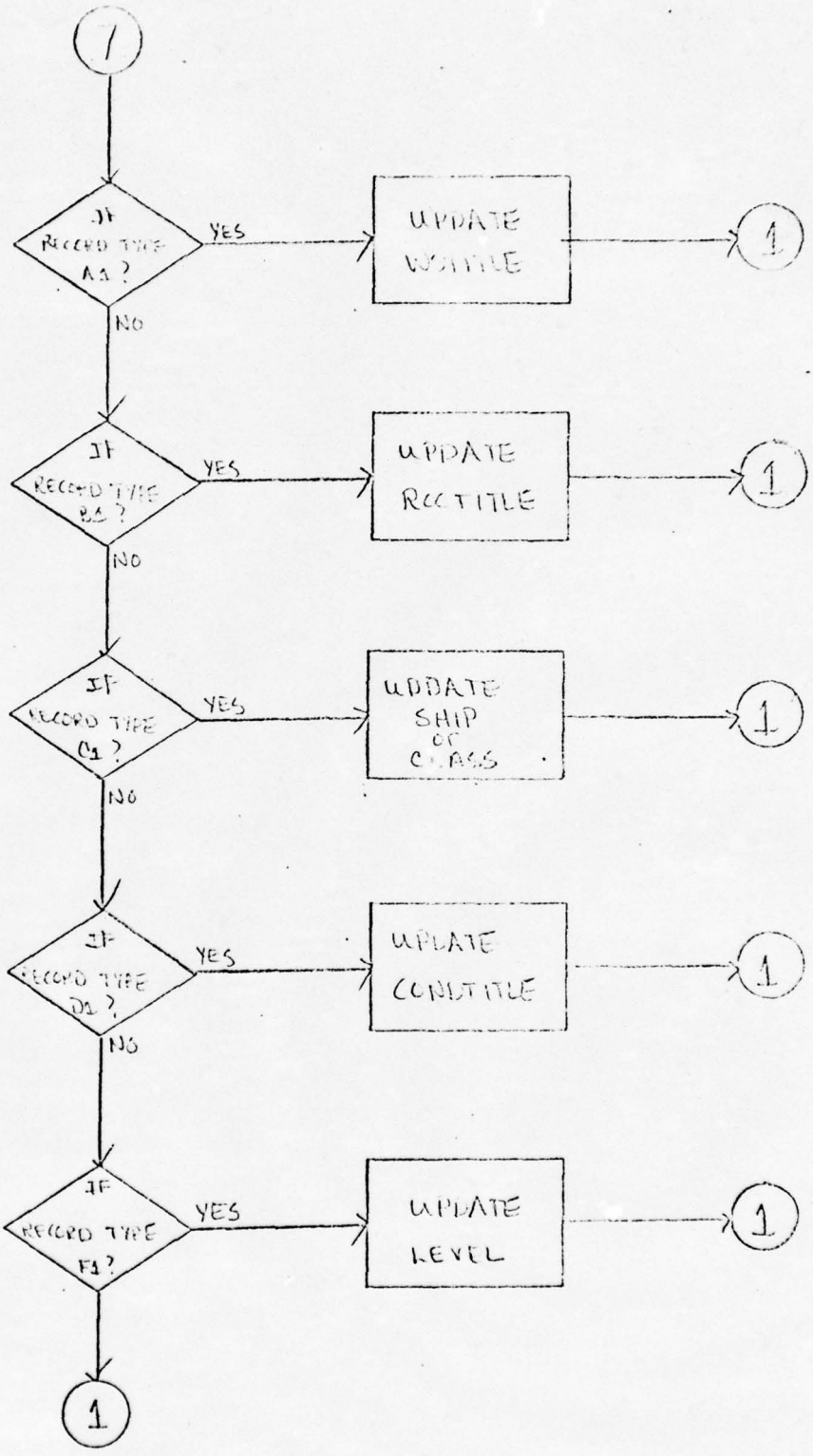

2
EDIT RECORD TYPE A1
ADD 1 TO READ-CTR
IF CARD-ERRORS ∅ ?
YES
7
NO
PRINT ERROR MESSAGES
1

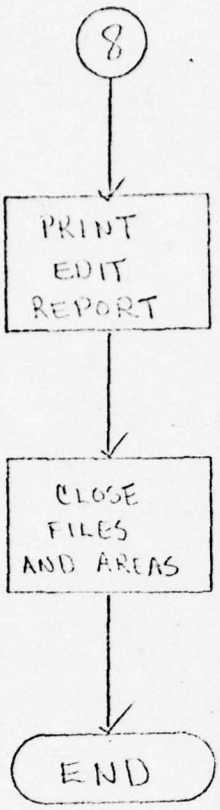

3
EDIT RECORD TYPE B1
ADD 1 TO READ-CTR
IF CARD-ERRORS ϕ ?
YES
7
NO
PRINT ERROR MESSAGES
1


4
EDIT RECORD TYPE C4
ADD 1 TO READ-CTR
IF CARD-ERRORS ≠ ?
YES
7
NO
PRINT ERROR MESSAGES
1


5
EDIT RECORD
TYPE D1
ADD 1
TO
READ-CTR
IF
CARD-ERRORS
= ?
YES
7
NO
PRINT
ERROR
MESSAGES
1


6
EDIT RECORD
TYPE F1
ADD 1
TO
READ-CTR
IF
CARD-ERRORS
= ?
YES
7
NO
PRINT
ERROR
MESSAGES
1


1
IF RECORD TYPE A1 ?
YES
UPDATE WPNTITLE
1
NO
IF RECORD TYPE B1 ?
YES
UPDATE RCCTITLE
1
NO
IF RECORD TYPE C1 ?
YES
UPDATE SHIP OR CLASS
1
NO
IF RECORD TYPE D1 ?
YES
UPLATE CONDTITLE
1
NO
IF RECORD TYPE F1 ?
YES
UPLATE LEVEL
1
1


8
PRINT
EDIT
REPORT
CLOSE
FILES
AND AREAS
END

APPENDIX D

MODULE NMWB01

Module Description

This module updates (adds, changes, and deletes) the following record types on the Ship ROC/Watchstation Module Database: WSTITLE, ROCTITLE, CLASS, SHIP, and CONDTITLE. The program was written under contract number N-00014-76-C-0473, effective 24 November 1975.

Support Software Environment.

a. This module of the NMRS must be written in standard Common Business Oriented Language (COBOL). This language is described in detail in the IBM OS Full American National Standard COBOL manual. No IBM extensions (designated by a shaded background in the COBOL manual) may be used in the programming of this module. The allowable exceptions are the CALL, CANCEL, USING and GOBACK statements and the use of the linkage section to transfer parameters during the execution of the CALL.

b. The general software environment is dependent upon the hardware configuration on which the NMRS is operating. The available software at NIH is listed in the NIH Computer Center Users Guide.

Inputs

Input to NMWB01 may be any or all of five different types of input transactions. The input file is described in the Characteristics paragraph below.

Outputs

Output from NMWB01 may be any or all of five different types of Ship/ROC Database records, namely WSTITLE, ROCTITLE, CLASS, SHIP, and CONDTITLE.

Working Storage

Working storage consists of report headers and detail line formats; tables used for accumulating totals, printing reports, and converting data; accumulators for various totals; and subscripts used in assessing table elements.

Characteristics

A. IT-REC is the card input transaction. The file is a concatenation of up to four individual files that are output from utility sorts.

B. Each input transaction is edited according to specifications for the particular record type associated with the transaction. If the record contains valid data, the transaction is used to update a particular Database record; otherwise the transaction is printed on an error report where the invalid card data is identified.

C. The data in the transaction header identifies the type of transaction, and the Database record type with which it is associated. The record key is used to either locate the Database record to be modified or deleted or place the Database record to be added. The text is used to modify or build a Database record.

D. All of the transaction data is static.

E. The input transaction file is a temporary disc file that requires one 3330 disc track per 156 input records.

F. The input files may be concatenated in any order. The sequence within the files is determined by utility sorts and will be listed below.

A1: Descending on function, ascending on W-A1-WSID.
B1: Descending on function, ascending on W-B1-SOCID.
C1: Ascending on card type, descending on function, Ascending on W-C1-CLASSID/W-C2-SHIPID.
D1: Descending on function, ascending on W-D1-CONDITION.

G. The Database Administrator should approve all changes, deletions, and additions to the Database submitted through this module. All the files updated by this module are used as "dictionary" files to validate data input to NMWB02 to update all other Database files. Actual restrictions built into the module are:

(1) A CLASS record may not be deleted until all of its member SHIP records have been deleted.

(2) A new CLASS and its new SHIP members may be added in the same run, but a CLASS and its SHIP members may not be deleted in the same run.

Data Environment

The purpose of W-MONTH-TABLE is to provide abbreviations for the twelve months in order to include the alpha month on report headings. The table values are hard-coded in the module. W-ERROR-MESSAGES holds comments that are printed on an error report to identify incorrect data on the input transactions. The subscript used to access the table is W-CRD-ERRS.

Program Logic

Note: Throughout the program whenever the database is accessed, the error status field is examined. When the value of the error status indicates a problem, an error routine is performed, edit totals are printed, and processing is ended. This logic will not be included in the description below.

Program processing begins with all files being opened, and database areas being made ready. Headers and print areas are initialized, and then file processing begins.

When an input transaction (will refer to input transaction as 'card') is read, all of the card data except card type is moved into Working Storage. The card type is then examined, and a branch is made to an appropriate edit routine.

If the card type is 'A1', the processing is as follows: An accumulator for A1 cards read is incremented. If the Major Functional Area is not numeric, an error message is stored. If Minor Functional Area is not numeric and equals spaces, the spaces are replaced with zeros. If it is not numeric and does not equal spaces, an error message is stored. If the sequence code equals spaces, they are replaced with zeros. If it is not numeric, an error message is stored. If the Minor Functional Area equals zeros and Seqence code does not equal zeros, an error message is stored. If the card function is 'D' the editing is complete and one of two paths may be followed: If there were no card errors to this point, there is a branch to the routine to delete WSTITLE records. If errors were found, there is a branch to an end of transaction routine. If card type is not 'D', editing continues. If the Watchstation Title is not present, an error message is stored. If the card function is not 'A', or 'C', or 'D' an error message is stored. Editing is now complete. Now if card errors occurred there is a branch to an end of transaction routine. If no errors occurred, there is either a branch to the change WSTITLE routine if function is 'C'; or the program drops into the add WSTITLE routine.

The processing to add a WSTITLE record is as follows: The WSTITLE card data is moved into the appropriate WSTITLE record fields in Working Storage and the record is stored. If the error status indicates a duplicate, an error message is stored and there is a branch to an end of transaction routine. If the record is added successfully, an accumulator for WSTITLE records added is incremented, the record and its database key are displayed, and there is a branch to an end of transaction routine.

The change routine for a WSTITLE record follows: There is an attempt to obtain the record to be changed. If the error status indicates that the record does not exist, an error message is stored and there is a branch to an end of transaction routine. If the record is found, card data is moved into the appropriate WSTITLE record fields in working storage and the record is modified.

If the operation is successful, an accumulator for WSTITLE records changed is incremented, the record and its database key are displayed, and there is a branch to an end of transaction routine.

The processing to delete a WSTITLE record is as follows: There is an attempt to obtain the record to be deleted. If the error status indicates that the record does not exist, an error message is stored and there is a branch to an end of transaction routine. If the record is found, it is erased. If the operation is successful, an accumulator for WSTITLE records deleted is incremented and the program drops into the end of transaction routine.

The end of transaction routine follows: If card errors were found, a routine to print error messages is performed. Then there is a branch back to the read paragraph.

If the card type is 'B1', the processing is as follows: An accumulator for B1 transactions read is incremented. If the ROC number is not numeric, an error message is stored. If the function is 'D', there is no more editing and one of two branches are made: If there were no card errors, there is a branch to the ROCTITLE delete routine; if there were errors there is a branch to an end of transaction routine. If the Mission Area is blank, an error message is stored. If the Mission Area is not alphabetic, an error message is stored. If the Operational Capability Code is missing, an error message is stored and there is a branch to a routine to edit the Required Functional Capability (RFC) Code. If the Operational Capability Code is numeric, but not right-justified, an error message is stored and there is a branch to a routine to edit the RFC code. If the Operational Capability code is right-justified, but not numeric, an error message is stored. If the RFC is numeric or equals spaces there is a branch to end of edit routine. If RFC code is left-justified and not numeric, an error message is stored and there is a branch to an end of edit routine. If the RFC code is right-justified and numeric there is a branch to an end of edit routine, otherwise an error message is stored.

If the card function is not 'A', or 'C', or 'D', an error message is stored. The editing is now complete. If card errors were found there is a branch to an end of edit routine. If the data was correct there is a branch to a change routine if the function is 'C', or the program drops into the add routine for ROCTITLE records.

The ROCTITLE record add routine follows: The card data is moved into the appropriate ROCTITLE record fields in Working Storage and the record is stored. If the error status indicates a duplicate, an error message is stored and there is a branch to an end of transaction routine. If the record is added successfully, an accumulator for ROCTITLE records added is incremented, the record and its database key are displayed and there is a branch to an end of transaction routine.

The logic for changing a ROCTITLE record is as follows: There is an attempt to obtain the record to be changed. If the error status indicates that the record does not exist, an error message is stored and there is a branch to an end of transaction routine. If the record is found, the card data is moved into the appropriate ROCTITLE record fields in Working Storage and the record is modified. If the operation is successful, an accumulator for ROCTITLE records changed is incremented, the record and its database key are displayed, and there is a branch to an end of edit routine.

The routine to delete a ROCTITLE record is as follows: There is an attempt to obtain the record to be deleted. If the error status indicates that the record does not exist, an error message is stored and there is a branch to an end of transaction routine. If the record is found, it is erased. If the operation is successful, an accumulator for ROCTITLE records deleted is incremented and the program drops into the end of transaction routine.

The end of transaction routine follows: If card errors were found, a routine to print error messages is performed. Then there is a branch back to the read paragraph.

If the record type is 'C2', the processing is as follows. An accumulator for C2 transactions read is incremented. If the ship code is not present, an error message is stored. If the class code is not numeric an error message is stored. If the card function is 'D' the editing is complete and one of two paths may be followed: If there were no card errors to this point, there is a branch to the routine to delete SHIP records. If errors were found, there is a branch to an end of transaction routine. If the card function is not 'D', editing continues. If identification is missing, an error message is stored. If the ship name is missing an error message is stored. If the card function is not 'A', or 'C', or 'D' an error message is stored. Editing is now complete. Now if card errors occurred there is a branch to an end of transaction routine. If no errors occurred, there is either a branch to the change SHIP routine if function is 'C', or the program drops into the add SHIP routine.

The processing to add a SHIP record is as follows: There is an attempt to obtain the CLASS owner record of the CLASS-SHIP set to which the SHIP is to be added. If the error status indicates that the record does not exist, an error message is stored and there is a branch to an end of transaction routine. If the record is found, the SHIP data from the card is moved into the appropriate SHIP fields in Working Storage and the new SHIP record is stored. If the error status indicates a duplicate, an error message is stored and there is a branch to an end of transaction routine. If the record is added successfully, an accumulator for SHIP record is incremented, the record and its database key are displayed, and there is a branch to an end of transaction routine.

The change routine for a SHIP record is as follows: There is an attempt to obtain the SHIP record to be changed. If the error status indicates the record does not exist, an error message is stored and there is a branch to an end of transaction routine. If the record is found, there is an attempt to obtain the CLASS owner record within the SHIP's CLASS-SHIP set. If the owner is found, it is compared to the class field on the card. If the two are not equal, an error message is stored and there is a branch to an end of transaction routine. If they are equal, the currency of the SHIP record to be changed is re-established and change processing continues. Card data is moved into the appropriate SHIP record fields in Working Storage and the record is modified. If the operation is successful, an accumulator for SHIP records changed is incremented, the record and its database key are displayed, and there is a branch to an end of transaction routine.

The delete routine for SHIP records follows: There is an attempt to obtain the SHIP record to be deleted. If the error status indicates the record does not exist, an error message is stored and there is a branch to an end of transaction routine. If the record is found, there is an attempt to obtain the CLASS owner record within the SHIP's CLASS-SHIP set. If the owner is found, it is compared to the class field on the card. If the two are not equal, an error message is stored and there is a branch to an end of transaction routine. If they are equal, the currency of the SHIP record is established and it is erased. If the operation is successful, an accumulator for SHIP records deleted is incremented and the program drops into the end of transaction routine.

The end of transaction routine follows: If card errors were found, a routine to print error messages is performed. Then there is a branch back to the read paragraph.

If the card type is 'C1' the processing is as follows: An accumulator for C1 cards read is incremented. If class code is not numeric, an error message is stored. If the function is 'D', there is no more editing and one of two branches is made: If there were no card errors, there is a branch to the CLASS delete routine; if there were errors there is a branch to an end of transaction routine. If function is not 'D', editing continues. If ship code is not present, an error message is stored. If card function is not 'A', or 'C', or 'D' an error message is stored. If card errors were found there is a branch to an end of edit routine. If the data was correct, there is a branch to a change routine if the function is 'C', or the program drops into the add routine for CLASS records.

The processing to add a CLASS record is as follows: The CLASS card data is moved into the appropriate CLASS record fields in Working Storage and the record is stored. If the error status indicates a duplicate, an error message is stored and there is a branch to an end of transaction routine. If the record is added

successfully, an accumulator for CLASS records added is incremented, the record and its database key are displayed, and there is a branch to an end of transaction routine.

The routine to change a CLASS record follows: There is an attempt to obtain the record to be changed. If the error status indicates that the record does not exist an error message is stored and there is a branch to an end of transaction routine. If the record is found, card data is moved into the appropriate record fields in Working Storage and the record is modified. If the operation is successful, an accumulator for CLASS records changed is incremented, the record and its database key are displayed, and there is a branch to an end of transaction routine.

The processing to delete a CLASS record is as follows: There is an attempt to obtain the record to be deleted. If the error status indicates that the record does not exist, an error message is stored and there is a branch to an end of transaction routine. If the record is found, it is erased. If the operation is successful, an accumulator for CLASS records deleted is incremented and the program drops into the end of transaction routine.

The end of transaction routine performs a routine to print error messages if the card had errors and then branches back to the read paragraph.

If the card type is 'D1' the processing is as follows: An accumulator for D1 records read is incremented. If the condition is spaces, an error message is stored. If function is 'D' and there is no error so far, there is a branch to a delete CONDTITLE records routine. If function is 'D' and data is incorrect, there is a branch to an end of transaction routine. If condition title is missing, an error message is stored. If card function is not 'A', 'C' or 'D' an error message is stored. The editing is now complete. If card errors were found there is a branch to an end of edit routine. If the data was correct, there is a branch to a change routine if the function is 'C'; or the program drops into the add routine for CONDTITLE records.

The CONDTITLE record add routine follows: The card data is moved into the appropriate CONDTITLE record fields in Working Storage and the record is stored. If the error status indicates a duplicate, an error message is stored and there is a branch to an end of transaction routine. If the record is added successfully, an accumulator for CONDTITLE records added is incremented, the record and its database key are displayed, and there is a branch to an end of transaction routine.

The logic for changing a CONDTITLE record is as follows: There is an attempt to obtain the record to be changed. If the error status indicates that the record does not exist, an error message is stored and there is a branch to an end of transaction routine. If the record is found, the card data is moved into the appropriate CONDTITLE record fields in Working Storage and the record is

modified. If the operation is successful, an accumulator for CONDTITLE records changed is incremented, the record and its database key are displayed, and there is a branch to an end of edit routine.

The processing to delete a CONDTITLE record is as follows: There is an attempt to obtain the record to be deleted. If the error status indicates that the record does not exist, an error message is stored and there is a branch to an end of transaction routine. If the record is found, it is erased. If the operation is successful, an accumulator for CONDTITLE records deleted is incremented and the program drops into the end of transaction routine.

The end of transaction routine performs a routine to print error messages if errors were found and then branches back to the read paragraph.

The database error routine is performed whenever an error status that indicates a problem is returned. The card being processed is displayed, along with the error status and an indicator that locates the paragraph where the problem occurred. It branches to the routine to print edit totals.

The following is the logic for the routine to print error messages: If the line count exceeds 46, a routine to begin a new page is performed. The input transaction is moved to the print line and is printed. The card is underlined and another line is printed with spaces. The line count is incremented. There is a loop to print the error messages: A subscript (W-ERR-IND) is incremented by one. If the subscript is greater than the number of errors that occurred (W-CRD-ERRS), the loop is ended. There is a routine performed to determine whether to go to a new page and print heading if necessary. An error message is moved from a table (W-PRT-ERR) to print line and is printed. The line count is incremented. Then there is a branch to the beginning of the loop which continues until all error messages are printed. Once out of the loop, an accumulator for card type errors is incremented, two lines of spaces are printed, the table that held messages and the area that held the card in error are cleared, and the subscript and the error counter are re-initialized to zero.

The header routine sets the line count back to zero and adds 1 to the page counter. It then prints headings on a new page by moving to the print line headers that are set up in Working Storage and printing them one by one.

The line check routine checks to see if the line count is greater than 46, and if it is, the header routine is performed.

The routine to print a line simply writes the line and then re-initializes the print line to spaces.

The routine to print edit totals loops through a table where the totals are stored (W-EDT-TABLE). Before the loop begins, headers are printed on a new page. The loop is as follows: A subscript (W-TYPE-IND) to identify the card type is incremented. If the subscript is greater than 5 (only five card types), the subscript is re-initialized to zero and there is a branch to an end of program routine. Another subscript (W-FLD-CTR) is incremented (for the totals). If the subscript is greater than 5, it is re-initialized to zero and there is a branch to print the line. The data from the table is moved to the print line and printed. Then there is a branch to the beginning of the loop. The loop continues until all the edit totals are printed.

The last paragraph closes all files, closes the database areas, and ends program processing.

PROGRAM FLOWCHART

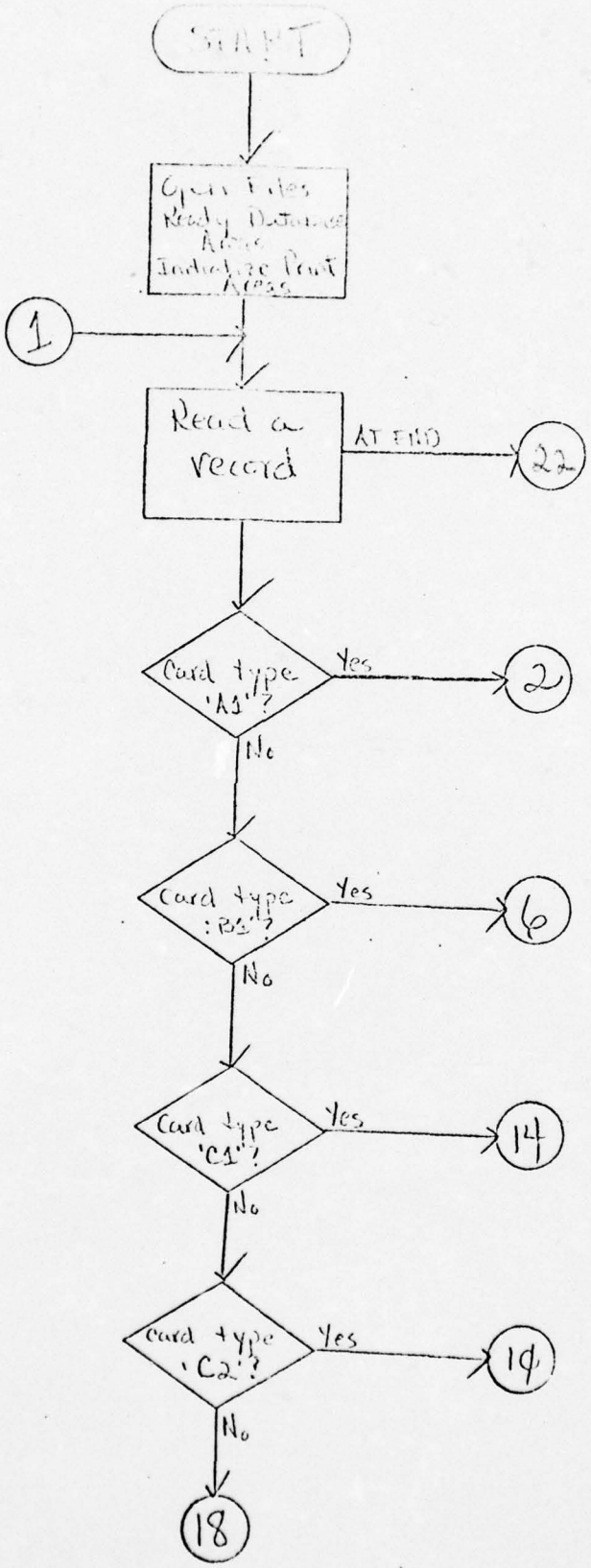
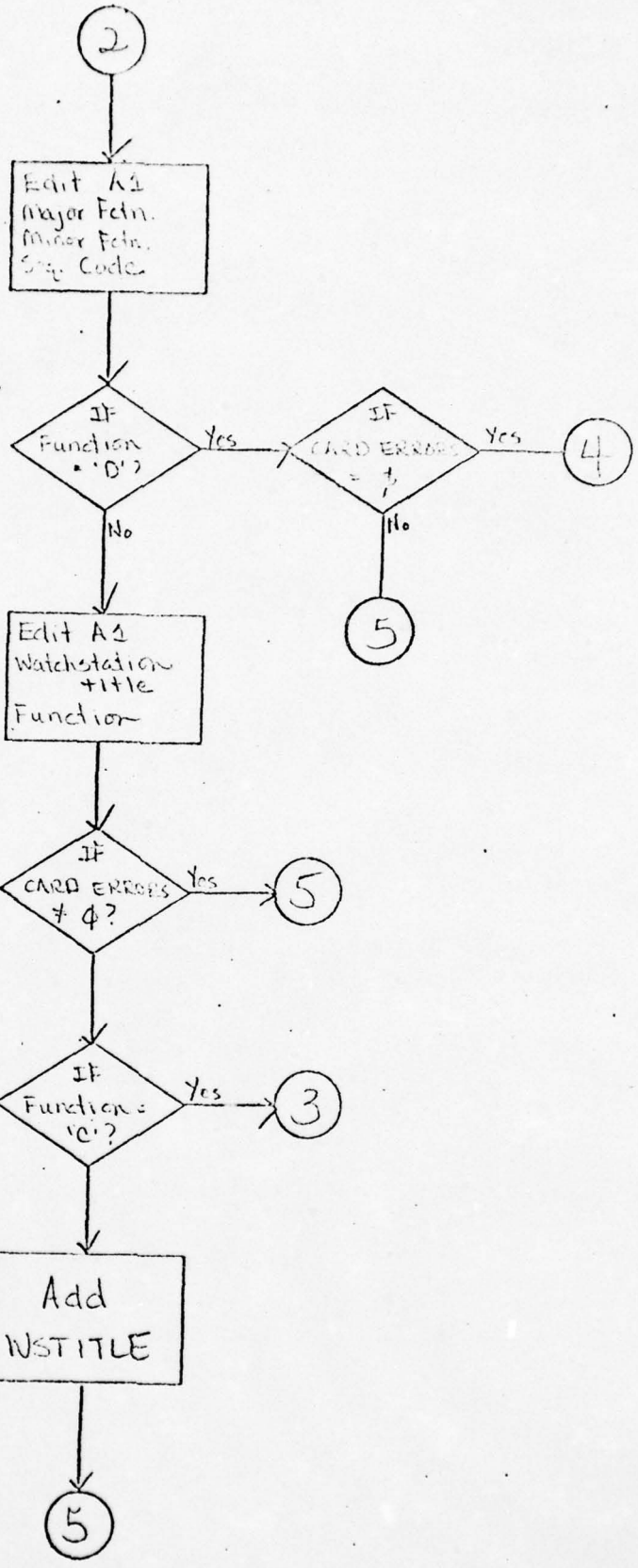
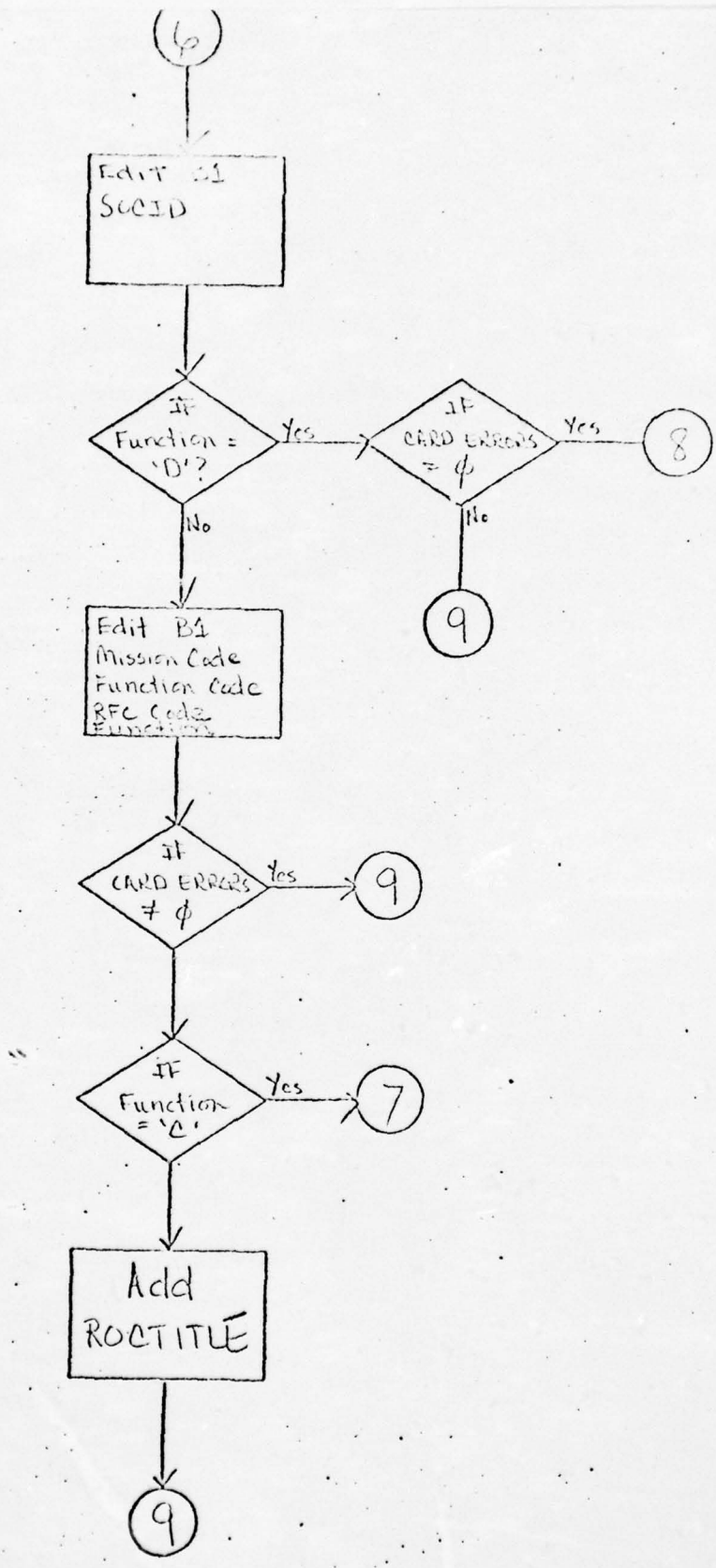
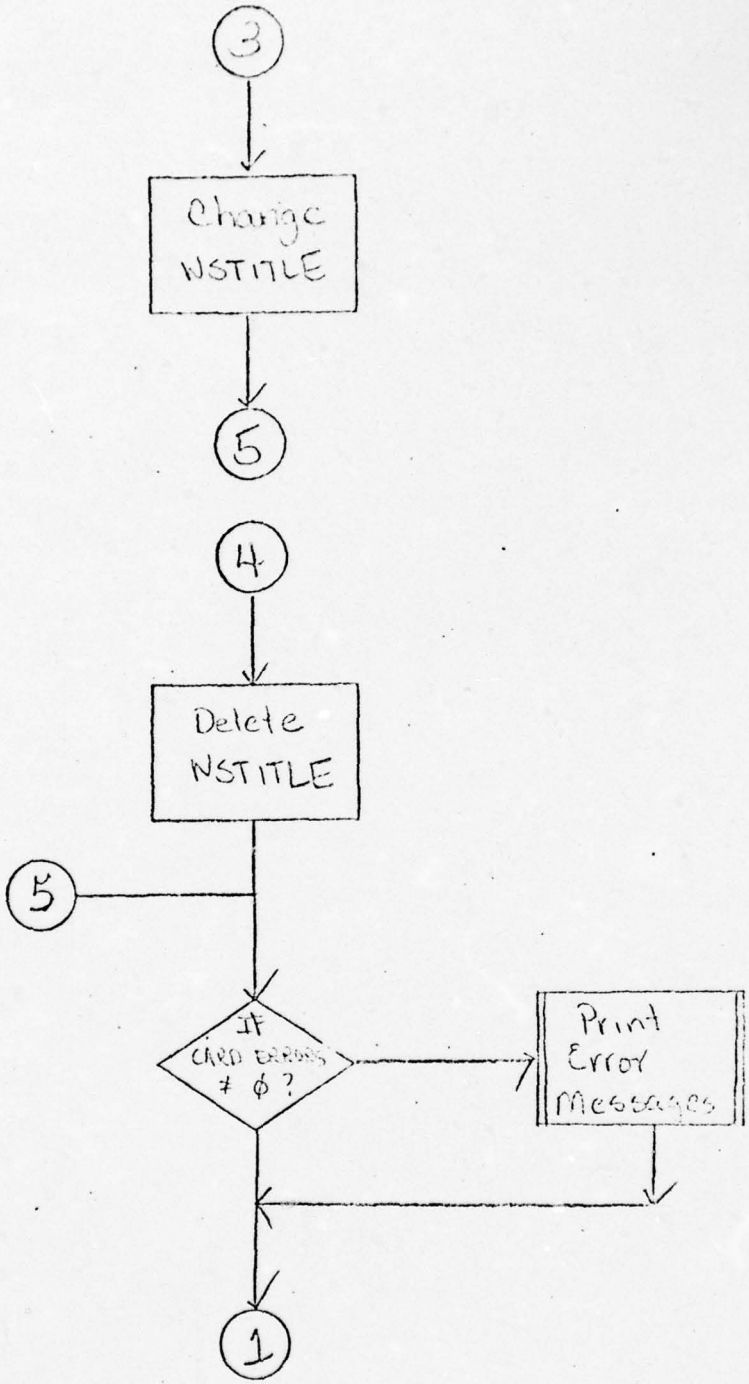


Figure D-1

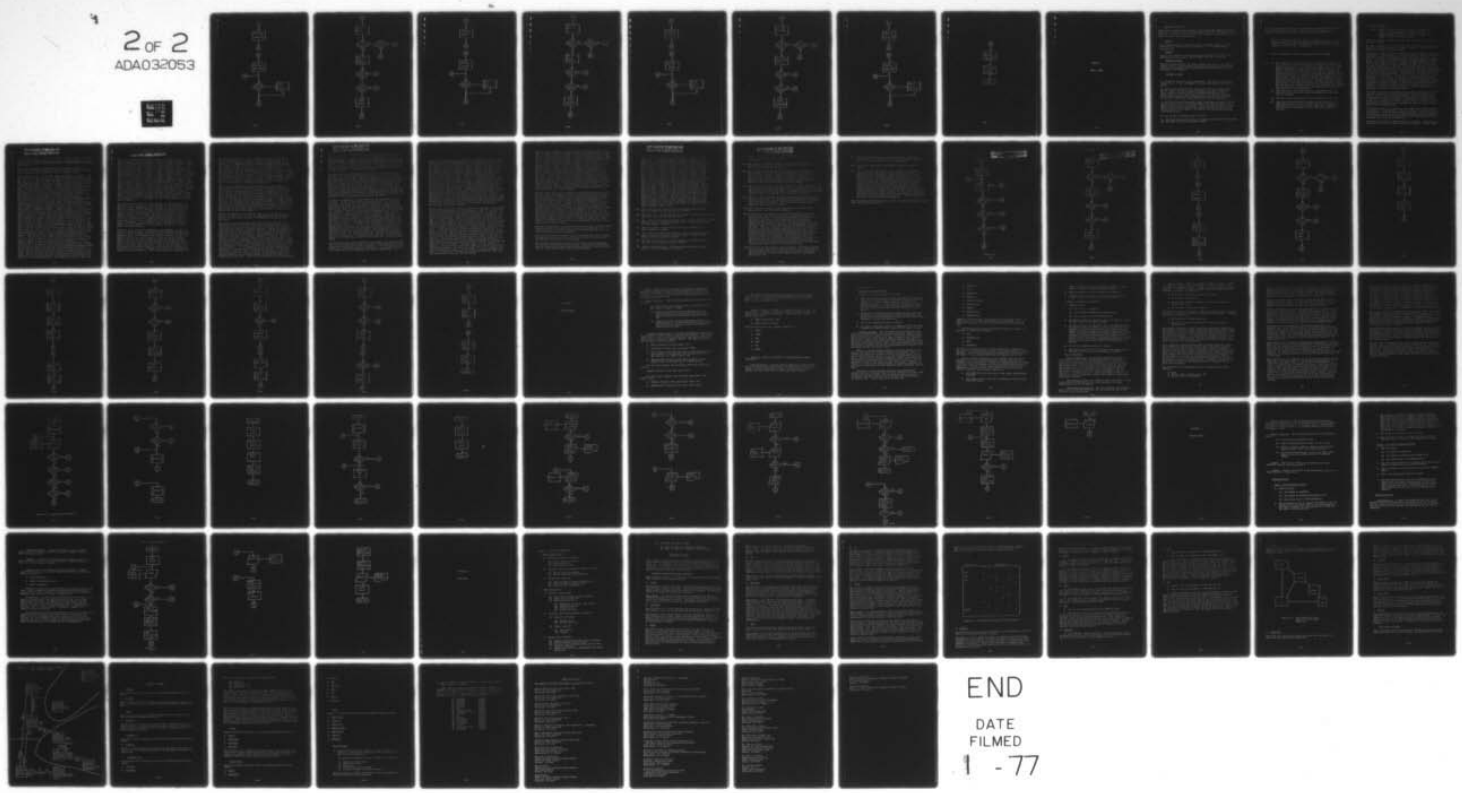

2
Edit A1
Major Fctn.
Minor Fctn.
Seq Code
IF
Function
= 'D'?
Yes
IF
CARD ERRORS
= 1
Yes
4
No
5
Edit A1
Watchstation
+title
Function
IF
CARD ERRORS
≠ 0?
Yes
5
IF
Function =
'C'?
Yes
3
Add
WSTITLE
5


6
Edit 21
SCCID
IF
Function =
'D'?
Yes
IF
CARD ERRORS
= ϕ
Yes
8
No
No
9
Edit B1
Mission Code
Function Code
RFC Code
IF
CARD ERRORS
≠ ϕ
Yes
9
IF
Function
= 'C'
Yes
7
Add
ROCTITLE
9

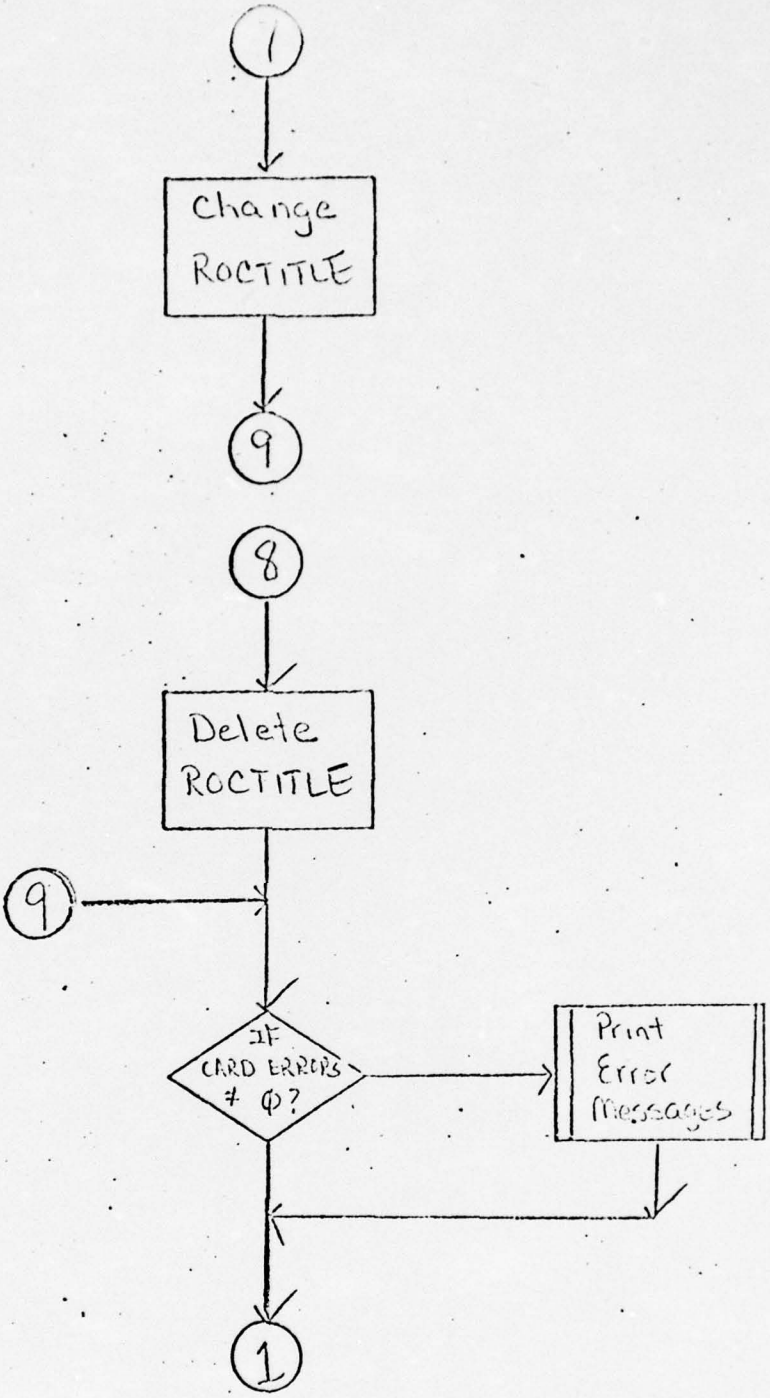

3
Change NSTITLE
5
4
Delete NSTITLE
5
IF CARD ERRORS ≠ ϕ ?
Print Error Messages
1

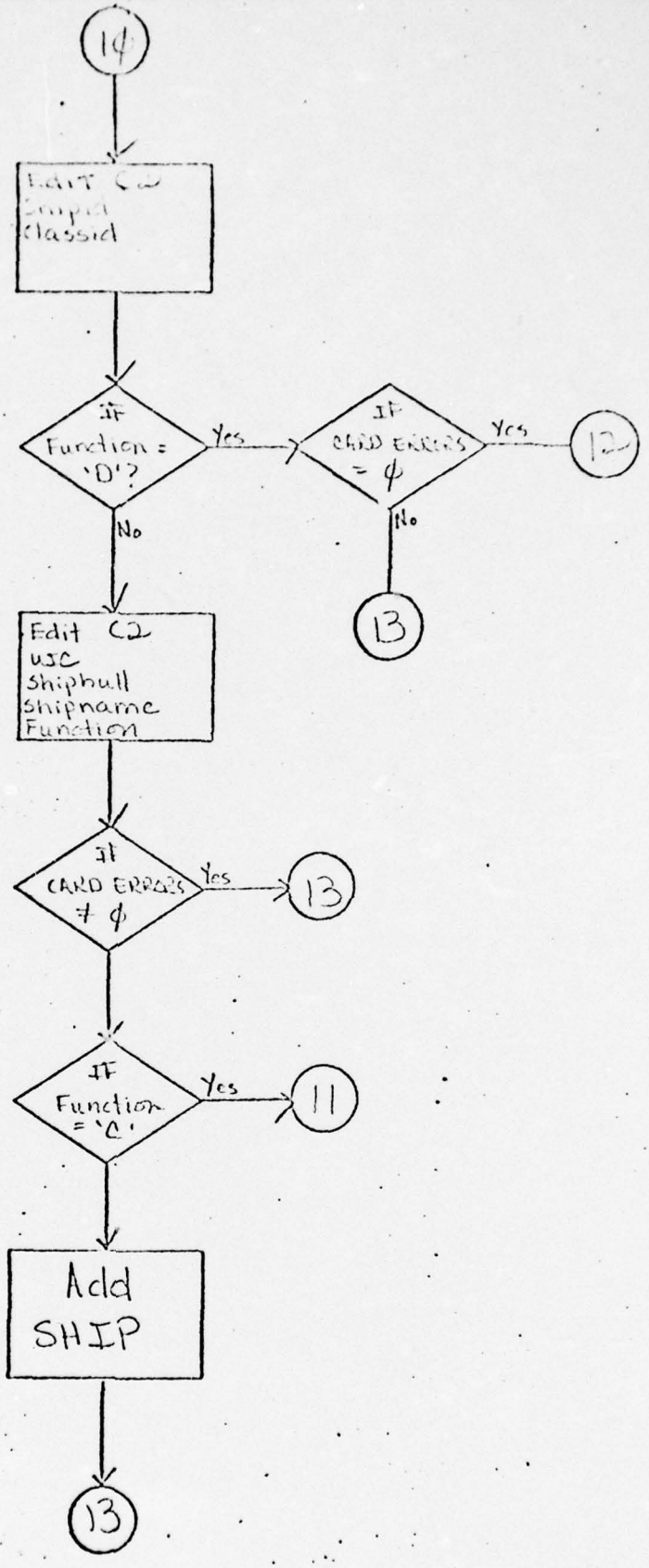
AD-A032 053 MANAGEMENT SCIENCE SYSTEMS INC FALLS CHURCH VA F/G 5/9
THE RELATIONSHIP OF MANPOWER AND OPERATIONAL TASKING FOR SHIPS:--ETC(U)
JUL 76 D R BENBENNICK, P P BRUCE, E W LULL N00014-76-C-0473
UNCLASSIFIED 357602 NL

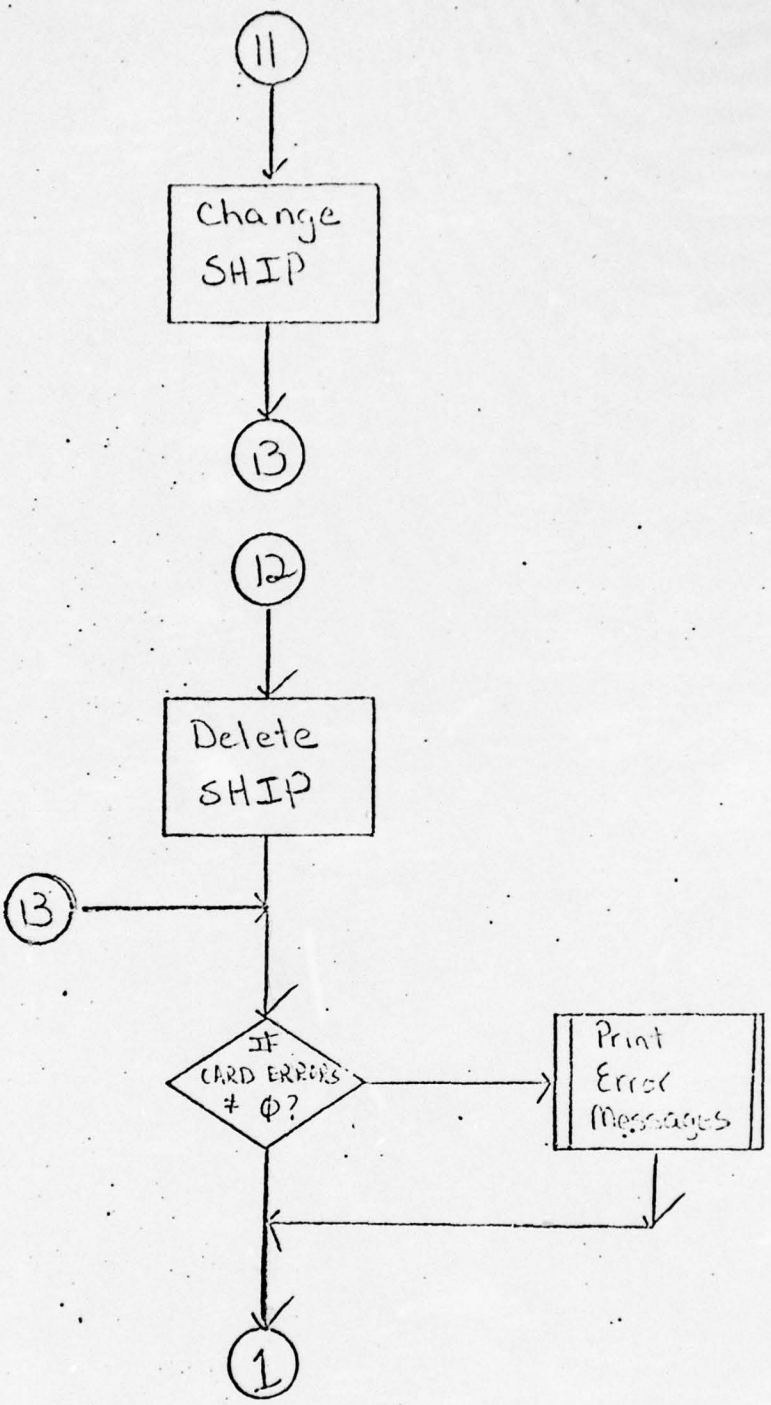
2 OF 2
ADA032053

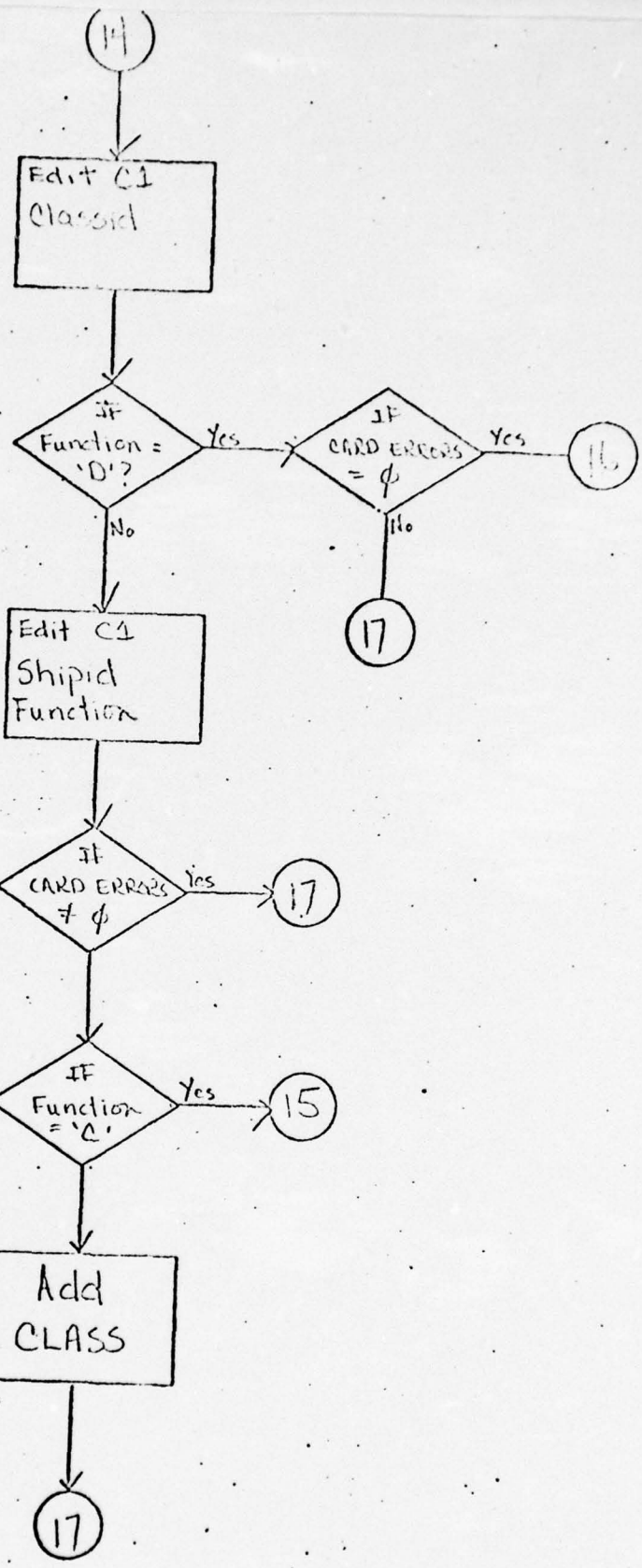


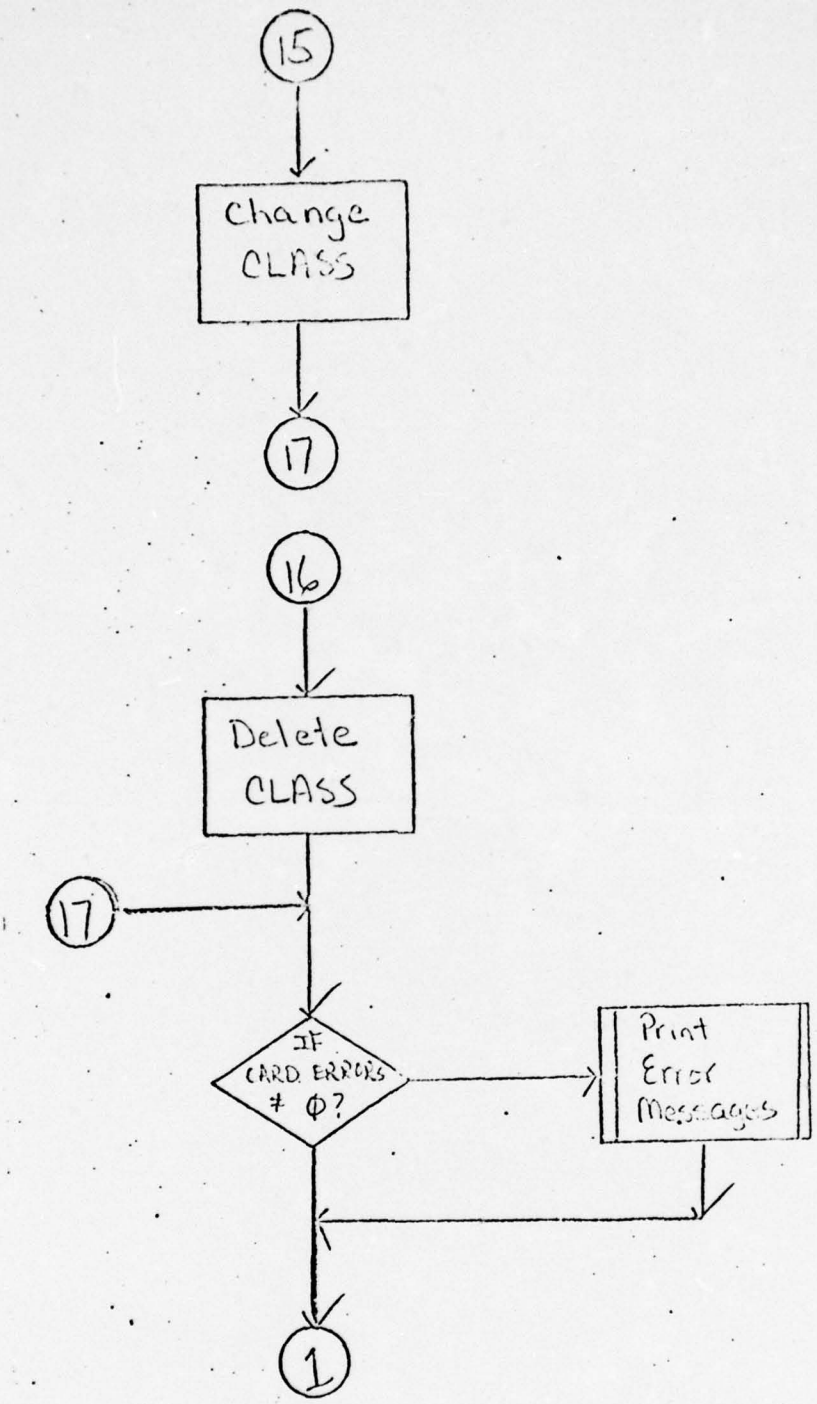
END
DATE FILMED
1 - 77

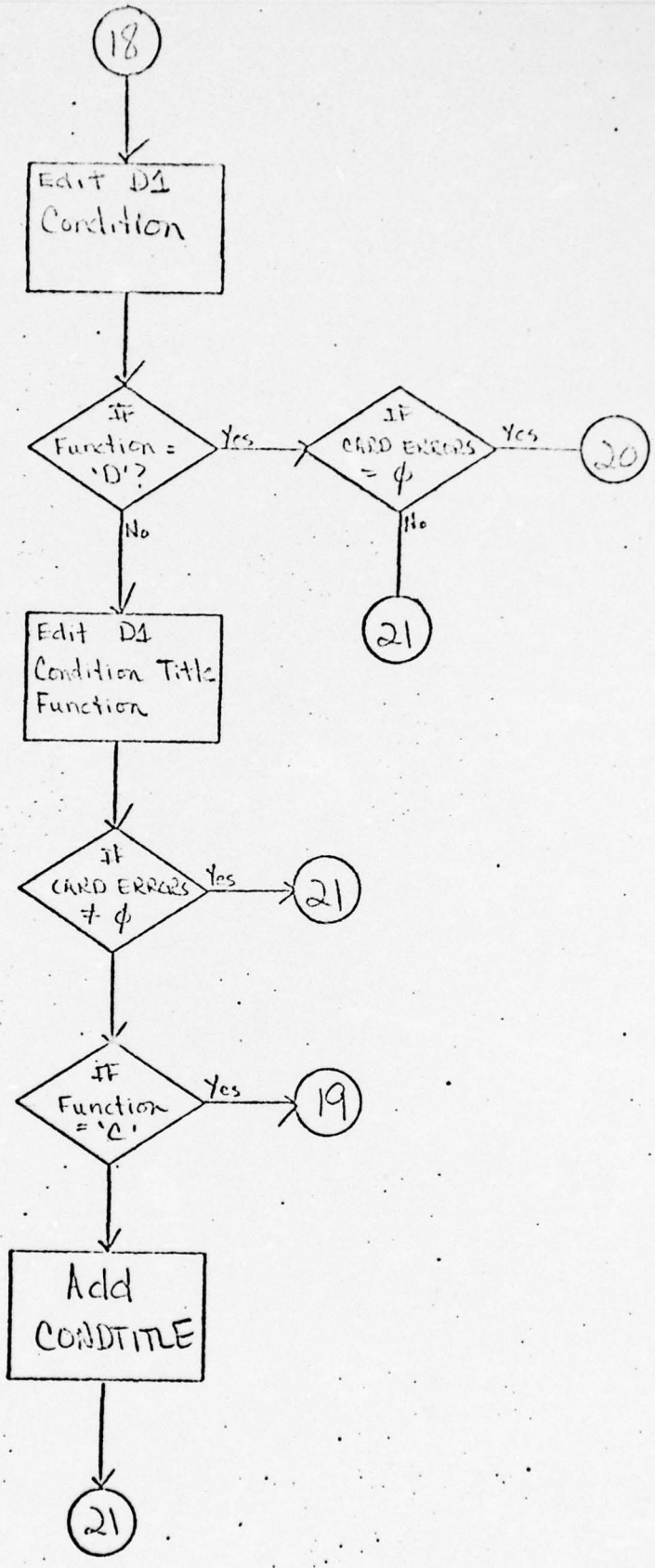

1
Change ROCTITLE
9
8
Delete ROCTITLE
9
IF CARD ERRORS ≠ Φ?
Print Error Messages
1

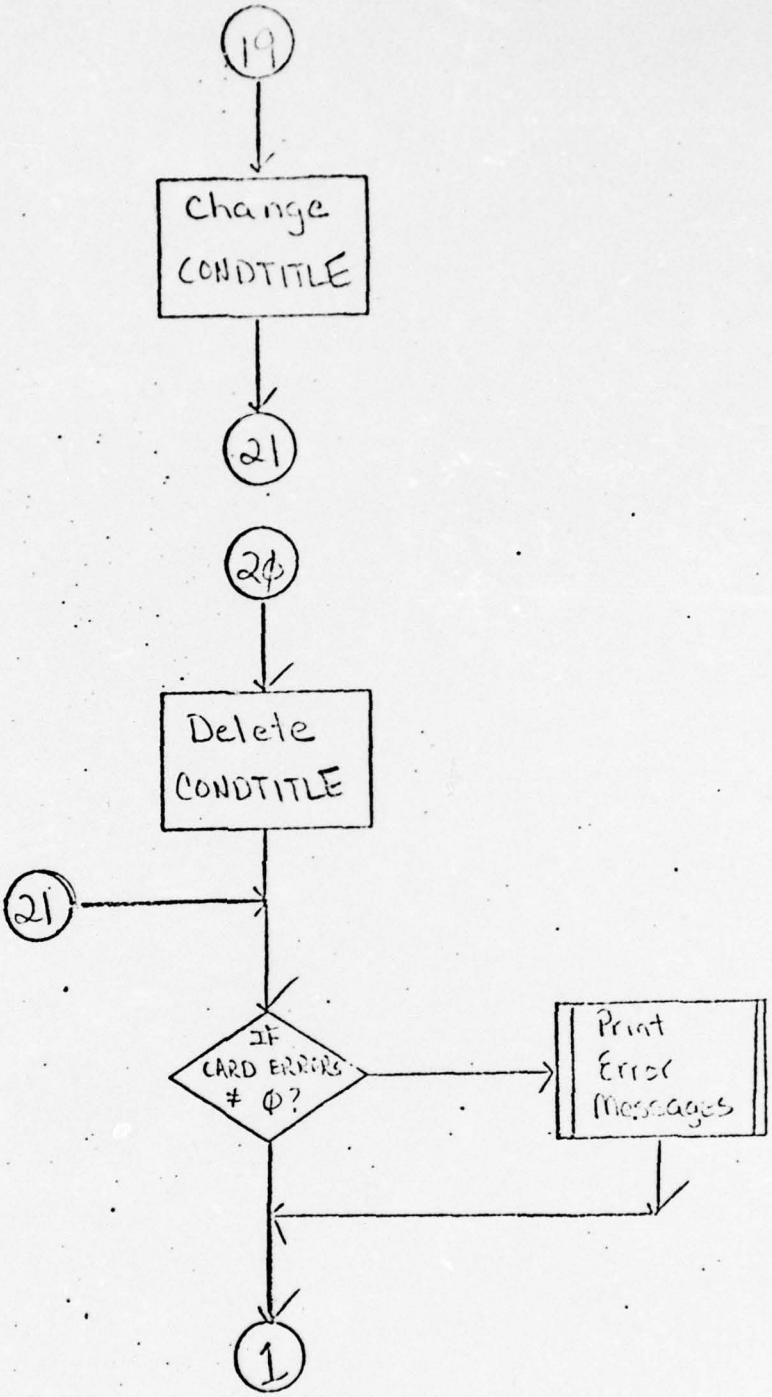

14
Edit C2
Empid
Classid
IF
Function =
'D'?
Yes
IF
CARD ERRORS
= ϕ
Yes
12
No
13
Edit C2
UIC
Shiphull
Shipname
Function
IF
CARD ERRORS
≠ ϕ
Yes
13
IF
Function
= 'C'
Yes
11
Add
SHIP
13

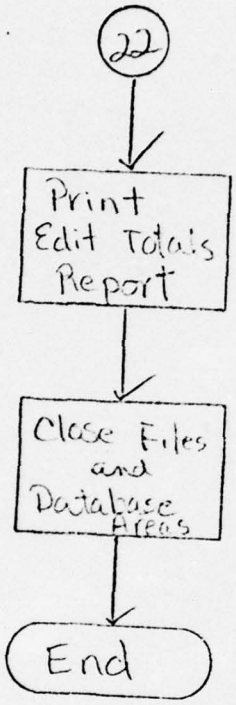

11
Change SHIP
13
12
Delete SHIP
13
IF CARD ERRORS ≠ Φ?
Print Error Messages
1


14
Edit C1
Classid
If
Function =
'D'?
Yes
If
CARD ERRORS
= ϕ
Yes
16
No
No
17
Edit C1
Shipid
Function
If
CARD ERRORS
≠ ϕ
Yes
17
If
Function
= 'C'
Yes
15
Add
CLASS
17


15
Change CLASS
17
16
Delete CLASS
17
IF CARD ERRORS ≠ ϕ?
Print Error Messages
1


18
Edit D1 Condition
IF Function = 'D'?
Yes
IF CARD ERRORS = ϕ
Yes
20
No
No
21
Edit D1 Condition Title Function
IF CARD ERRORS ≠ ϕ
Yes
21
IF Function = 'C'
Yes
19
Add CONDTITLE
21


19
Change
CONDTITLE
21
24
Delete
CONDTITLE
21
IF
CARD ERRORS
≠ Φ?
Print
Error
Messages
1


22
Print Edit Totals Report
Close Files and Database Areas
End

APPENDIX E

MODULE NMWB02

Module Description

This module updates (adds, changes, or deletes) the following record types on the ship ROC/Watchstation Module Database: WS, WSROC, LEVEL, and ROC. The program was written under contract number N-00014-75-C-0473, effective 24 November 1975.

Inputs

Input to NMWB02 may be any or all of six different types of input transactions. The input file is described in the Characteristics paragraph below.

Outputs

Output from NMWB02 may be any or all of four different types of SHIP/ROC database records, namely LEVEL, ROC, WS, and WSROC.

Working Storage

Working storage consists of report headers and detail line formats; tables used for accumulating totals, printing reports, and converting data; accumulators for various totals; and subscripts used in accessing table elements.

Characteristics

A. IT-REC is the card input transaction. The file is a concatenation of up to four individual files that are output from utility sorts.

B. Each input transaction is edited according to specifications for the particular record type associated with the transaction. These requirements are stated in the Module Specifications. If the record contains valid data, the transaction is used to update a particular database record; otherwise the transaction is printed on an error report where the invalid card data is identified.

C. The data in the transaction header identifies the type of transaction, the function of the transaction, and the database record type with which it is associated. The record key is used to either locate the database record to be modified or deleted or place the database record to be added. The text is used to modify or build a database record.

D. All of the transaction data is static.

E. The input transaction file is a temporary disc file that requires one 3330 disc track per 156 input records.

F. The individual files must be concatenated in the following order: F1, G1 and H1, G5 and H5, J1. The sequence within the individual files is determined by utility sorts and will be listed below.

F1: Descending on function, ascending on LEVELKEY. G1 and H1: Descending on card type, descending on function, ascending on W-H1-WSKEY. G5 and H5: Descending on card type, descending on function, ascending on W-H5-WSROC-KEY.

J1: Descending on function, ascending on W-J1-ROCKEY.

G. (1) When changing a WS record that has a duplicate(s) within a set, the user must be sure that the complete WS is changed. Only one change card may be submitted. This card will change the WS record that is the owner of a WS-WSROC set, if one exists. In order to change other duplicates or to have them remain as they are, add cards must be submitted containing the appropriate data. Example: Watchstation 0100100010 occurs three times within the CLASS-WS set that has class 001 as the owner record. The user wants to change the watchstander requirements for one occurrence of the watchstation and have the other duplicates remain unchanged. The following cards must be submitted: One change card with the new watchstander data, and two add cards containing the data currently on the database for the duplicate occurrences of the watchstation which the user wishes to remain unchanged.

(2) When deleting a WS record that has a duplicate(s) in the same set, all occurrences of that watchstation in that set will be deleted.

(3) The same record may be deleted and added back in one run.

(4) When modifying WS and LEVEL records on the database, the transaction that will be used must contain exactly the data that is to be on the database after the modification. The particular field that is to be changed is <u>not</u> sufficient.

4.6 Program Logic

Note: Throughout the program whenever the database is accessed, the error status is checked. When the error status indicates a problem, an error routine is performed and processing is ended. This logic will not be included below.

Files are opened, database areas are made available, and print areas are initialized.

An input transaction is read and the input data is stored in Working Storage. The transaction's card type is checked, and a branch is made to the appropriate edit routine.

If the card type is 'F1', the processing is as follows: An accumulator for 'F1' transactions read is incremented. If the level identification on the card is missing, an error message is stored in a table. A routine is performed to edit the ship identifier on the card. If the UIC field is not numeric an error message is stored. If the UIC field is present, a routine is performed to find the UIC in the SHIP file on the database. If the UIC is not found, an error message will be stored. If the card function is 'D' and no errors have occurred thus far, a branch is made to a delete routine. If the card function is 'D' and errors have been found, there is a branch to an end of transaction routine. If the card function is not 'D', editing continues. If the level name is missing, an error message is stored. If the update date is missing, or is not numeric, an error message is stored. A routine that edits card function for 'A', 'C', or 'D' is performed and if the function is not valid, an error message will be stored. At this point, if any errors have occurred, there is a branch to an end of transaction routine. If no errors have occurred and card function is 'C', there is a branch to a change routine. Otherwise the program falls into the add routine for a LEVEL record.

The SHIP owner of the SHIP-LEVEL set is found. Card data is moved into the appropriate LEVEL record fields and the new record is stored. If the error status indicates that an identical record already exists, an error message is stored and there is a branch to an end of transaction routine. If the record is added successfully, an accumulator for LEVEL records added is incremented, the record and its database key are displayed, and there is a branch back to the read paragraph.

The Change routine is as follows: The LEVEL record to be changed is obtained. If the record is found, card data is moved into the LEVEL record fields and the record is modified. If it is modified successfully, an accumulator for LEVEL records changed is incremented, the record and its database key are displayed, and there is a branch back to the read paragraph.

The routine to delete a LEVEL record is as follows: The ship and level are used to obtain the record to be deleted. If the record

is not found, an error message is stored and there is a branch to an end of transaction routine. If the record is found successfully, it is deleted. If the delete is successful, an accumulator for LEVEL records deleted is incremented and there is a branch back to the read paragraph.

The end of transaction paragraph for a LEVEL record adds to an accumulator for 'F1' cards with errors, performs a routine to print error messages, and branches back to the read paragraph.

If the card type is 'G1' or 'H1', the processing is as follows: An accumulator for 'G1' and 'H1' transactions read is incremented. If the card type is 'H1' the class on the card is moved to a hold area and there is a routine performed to edit class. If the card type is 'G1', the ship code on the card is moved to a hold area and there is a routine performed to edit UIC. If the class is found to be numeric on the 'H1' transaction, a routine to find the class in the CLASS file is performed. If the UIC on the 'G1' transaction is present, a routine to find the UIC in the SHIP file is performed. The watchstation is moved to a hold area and an edit routine is performed. If the watchstation is numeric a routine to find it in the WSTITLE file is performed. At this point the editing is finished if the card function is 'D', and one of the following occurs: If no card errors were found, there is a branch to the delete WS routine; if card errors were found, there is a branch to an end of transaction routine. If the function is not 'D', editing continues. If the WSMANNING field is not equal to either 'L', 'M', 'X', 'C', or space an error message is stored. There is a loop to edit the conditions: A condition counter is incremented by 1. If it exceeds 7, there is a branch out of the loop. If the condition is spaces, spaces are moved to a hold area and there is a branch to the beginning of the loop. If the condition is not spaces, but the hold area for conditions is (indicating a prior condition with spaces), an error message is stored and there is a branch out of the loop. The condition is looked up in the CONDTITLE file. It is not found, an error message is stored. There is a branch to the beginning of the loop. Once out of the loop, the condition counter is reset to zero and the condition hold area is reset to '99'. Next the OEC is checked for 'E' or 'O' and branches are made to the respective edit routines for either enlisted or officer. If the OEC is neither of the above, an error message is stored. If the enlisted Rating Designation and Associated Rating fields are both non-blank, an error message is stored. The enlisted edit converts an enlisted paygrade of 'AR', 'SR', or 'FR' to 'AR'; 'AA', 'SA', 'FA' to 'AA'; 'AN', 'SN' or 'FN' to ' N'. There is then a loop to convert the enlisted paygrades to numeric codes: An enlisted counter is incremented by 1. If the counter exceeds 9 (only 9 valid values), an error message is stored and there is a branch out of the loop. The pay grade from the card is compared to the pay grade in the table (ENLISTED-CONVERSION-TABLE). If the two are equal, the paygrade is overlayed with the current value of the enlisted counter and there is a branch out of the loop. If the two are not equal, there is a branch to the beginning of the loop. Once out of the loop, the enlisted counter is reset to zero and there is a branch to the next edit routine. The first edit for officers is rating designation. Valid values are spaces, 'OFF', or

numerics. If the rating designation is other than these, an error message is stored. If the officer Rating Designation and Associated Rating fields are both non-blank, an error message is stored. There is a loop to convert the pay grade to numerics: An officer counter is incremented by 1. If it exceeds 14 (only 14 valid values), an error message is stored and there is a branch out of the loop. The pay grade on the card is compared to one in the table (W-OFFICER-CONVERSION-TABLE). If the two are equal, the paygrade on the card is overlayed by the officer counter and there is a branch out of the loop. If the two are not equal, there is a branch back to the beginning of the loop. Once out of the loop, the officer counter is reset to zero. The field holding the number of watchstanders is edited next. If the field contains spaces, it is converted to '01'. If the field does not hold a value in the range '01' - '09', an error message is stored. If the OMWPERWEEK field is not numeric, an error message is stored. If one of either the Searchkey or the organizational code is not present, an error message is stored. If the Searchkey is present and is not numeric, an error message is stored. Next, a routine is performed that edits the card function, and the editing is complete. If any card errors were found, there is a branch to an end of transaction routine. If there were no errors, there is a branch to a change routine. If function is 'C', or the program drops into an add routine.

For card types 'G1' and 'H1' with function 'A' the processing is as follows: If card type is 'G1' the UIC on the card is used to obtain the SHIP record owner of the SHIP-WS set to which the new WS is to belong. If the card type is 'H1' the class on the card is used to obtain the CLASS owner record of the CLASS-WS set to which the new WS is to belong. The data from the card is moved into the appropriate WS record fields in Working Storage and the record is stored. If it is stored successfully it is connected to either a CLASS-WS set or a SHIP-WS set depending on the card type being 'H1' or 'G1', respectively. If the WS is connected to a set successfully, an accumulator for WS records added is incremented, the database key is displayed, and there is a branch back to the read routine.

To change a WS record the processing is as follows: If the key field on the card equals a hold area for that field and the card types are equal (indicating a duplicate WS record has already been changed), there is a branch to the 'Add WS' routine. If the card type is 'G1' the UIC field is used to obtain the SHIP record that is owner of the set to which the WS to be changed belongs, otherwise the class field is used to obtain the respective CLASS owner. The counter for errors is stored and the watchstation on the card is moved into a hold area. Then if the card type is 'G1' a routine is performed to locate the WS record to be changed via the SHIP-WS set otherwise a routine is performed to locate the WS record via the CLASS-WS set. After this routine is performed, if

an error occurred, there is a branch to an end of edit routine. If the WS record is located, its database key is stored. Then there is a routine performed to locate and process duplicate WS records. If a duplicate record is found and modified, a routine is performed to delete the first WS record obtained and there is a branch out of the change routine. If no duplicates are found the first WS record is re-obtained by using its database key, card data is moved into the appropriate WS record fields in Working Storage and the record is modified. If the operation is successful, an accumulator for WS records changed is incremented, the record modified is displayed along with its database key, and the WSKEY is stored. Once out of the change routine, the area to indicate duplicates changed is cleared, the duplicate counter is reset to zero and there is a branch back to the read paragraph.

To delete a WS record the following occurs: If the card type is 'G1' the SHIP owner of the WS record is obtained, otherwise the CLASS owner of the WS record is obtained. The card error counter is stored and the watchstation is moved to a hold area. If the card type is 'G1' a routine is performed to locate the WS record to be deleted via the SHIP-WS set; otherwise, a routine is performed to locate the record via the CLASS-WS set. If an error occurs, there is a branch to an end of transaction routine. If the record is found, its database key is stored. Then there is a routine performed to locate and process any duplicate WS records. The WS record whose key was stored is re-obtained and deleted. If the delete is successful, the counter for duplicates is set back to zero, an accumulator for WS records deleted is incremented, and there is a branch back to the read paragraph.

The end of transaction routine for card types 'G1' and 'H1' add to an accumulator for 'G1' and 'H1' cards with errors, performs a routine to print error messages, and branches back to the read paragraph.

The processing for card types 'G5' and 'H5' is as follows: An accumulator for 'G5' and 'H5' cards read is incremented. If the card type is 'H5' the class is moved to a hold area and a routine is performed to edit it, otherwise the UIC is moved to a hold area and there is a routine performed to edit it. If the field holding class is numeric; if card type is 'H5' a routine is performed to find the class in the CLASS file. The watchstation is moved to a hold area and there is a routine performed to edit it. If it is numeric, a performed routine looks it up in the WSTITLE file. The individual ROCs to be added or deleted are edited in a loop. There are three groups of four ROCs each and the loop proceeds as follows: A counter for 'group' is incremented by 1. If it exceeds 3, there is a branch out of the loop. A counter for ROCs within a group is incremented by 1. If it exceeds 4, it is re-initialized to zero and there is a branch to the beginning of the loop. If the ROC is spaces, spaces are moved to a hold area and there is a branch to the paragraph that edits one group. If the ROC is not spaces, but the hold area is, an error message is stored and there is a branch out of the loop. If the ROC is not numeric, an error message

is stored and there is a branch back to the beginning of the group edit. If the ROC is numeric, a routine is performed to locate it in the ROCTITLE file. Then there is a branch back to the beginning of the group edit. Once out of the loop, the counters for groups and ROCs within a group are set to zero, and the hold area is given the value '9999'. A routine to edit card function is performed. A second edit on function stores an error message if card function is 'C'. At this point, editing is complete. If errors occurred there is either a branch to the delete WSROCs because function is 'D', or the program drops into the add WSROC routine.

To add a WSROC(s) the following takes place: If the card type is 'G5' the UIC is used to obtain the SHIP owner of the WS to which the WSROCs are to be added; otherwise the class is used to obtain the CLASS owner of the WS to which the WSROCs are to be added. The error counter is stored and the watchstation is moved to a hold area. Then if the card type is 'G5', the WS owner of the WS-WSROC set to which the WSROCs are to be added is located via the SHIP-WS set; otherwise it is located via the CLASS-WS set. If an error occurs, there is a branch to an end of transaction routine. If the WS record is found, its database key is stored. A routine is performed that locates duplicate WS records. If no duplicates are found, the WS record whose database key is in hold is re-obtained and there is a branch to the paragraph that stores the new WSROCs.

If there is a duplicate, its WS-WSROC set is checked to see if it is empty. If it is, the duplicate counter is reset to zero and there is a branch back to find another duplicate. If the set is not empty, the WSROCs will be added to this set. There is a loop used to actually add the WSROCs that uses two subscripts because of the way the ROCs are set up on the transaction. The loop processing is as follows: The first subscript is incremented by 1. If it is greater than 3, there is a branch out of the loop. The second subscript is incremented by 1. If it exceeds 4, it is reset to zero and there is a branch back to the beginning of the loop. If the field is spaces, there is a branch out of the loop; otherwise the ROC is moved into the appropriate WSROC field in Working Storage and is stored. If the error status indicates a duplicate, an error message is stored, and there is a branch back to the second paragraph in the loop. If the add is successful, an accumulator for WSROCs added is incremented, the WSROC and its database key are displayed and there is a branch back to the beginning of the loop. Once out of the loop, both subscripts and the duplicate counter are set to zero. If any erros occurred, a routine is performed to print error messages. An accumulator for 'G5' and 'H5' add transactions is incremented, and there is a branch back to the read paragraph.

The processing for deleting WSROCs is as follows: The logic is the same as for adding WSROCs until the point of handling WS duplicates. A routine is performed to find a duplicate. If a duplicate is not found, the WS whose database key is stored is re-obtained and there is a branch out of the paragraph that handles duplicates. If a

duplicate is found and its WS-WSROC set is non-empty, the duplicate counter is reset to zero and there is a branch back to the beginning of the routine to handle duplicates. If no duplicates are found, the original WS record's WS-WSROC set is checked. If it is empty (no WSROCs to delete), an error message is stored and there is a branch to an end of transaction routine. The routine that deletes the WSROCs loops through the ROCs on the transactions that are divided into three groups with 4 ROCs per group: The group subscript is incremented by one. If it exceeds three there is a branch to get out of the loop. The second subscript is incremented by one. If it is greater than 4, it is reset to zero and there is a branch to the beginning of the loop. If the ROC field is blank, there is a branch to get out of the loop. Otherwise the ROC is used to obtain the WSROC. If it is not found an error message is stored and there is a branch to the second paragraph of the loop. If it is found, it is deleted. If the delete is successful, an accumulator for WSROCs deleted is incremented and there is a branch to the second paragraph of the loop. Once out of the loop, both subscripts and the duplicate counter are set to zero. If errors occurred, there is a routine performed to print error messages. An accumulator for 'G5' and 'H5' delete transactions read is incremented and there is a branch to the read paragraph. The end of transaction routine adds to an accumulator for 'G5' and 'H5' cards with errors, a routine to print error messages is performed, and there is a branch back to the read paragraph.

If the card type is 'J1', the processing is as follows: An accumulator for J1 transactions read is incremented. The UIC is moved to a hold area and a routine to edit it is performed. If the UIC is present, a routine to locate it in the SHIP file is performed. If the level is blank, an error message is stored. A routine to edit function is performed. Then, if the function is 'C' an error message is stored. There are three groups of ROCs on the transaction with four ROCs in each group. They are edited in a loop: The group subscript is incremented. If it exceeds 3 there is a branch to get out of the loop. The ROC subscript is incremented. If it exceeds 4, it is reset to zero and there is a branch to the beginning of the loop. If the ROC is blank, spaces are moved to a hold area and there is a branch back to the second paragraph of the loop. If the ROC is not spaces and the hold area is, an error message is stored. If the ROC is not numeric, an error message is stored and there is a branch back to the second paragraph of the loop. If the ROC is numeric, a routine is performed to look it up in the ROCTITLE file. Then there is a branch back to the second paragraph of the loop. Once out of the loop, the two subscripts are set to zero and the ROC hold area is set to '9999'. At this point, the 'J1' editing is complete. If card errors were found, there is a branch to an end of transaction routine. If errors are found there is a branch to the end of transaction routine. If no errors are found and card function is 'D' there is a branch to the delete ROC routine, otherwise the program drops into the add ROC routine.

To add a ROC the processing is as follows: The UIC and level are used to locate the LEVEL record owner of the LEVEL-ROC set to which the ROCs are to be added. If it is not found, an error message is stored and there is a branch to an end of transaction routine. The ROCs are added by a loop routine: A group subscript is incremented. If it exceeds 3, there is a branch to get out of the loop. A second subscript for ROCs within a group is incremented. If it is greater than 4, it is reset to zero and there is a branch to the beginning of the loop. If the ROC field is blank, there is a branch to get out of the loop. The ROCTITLE record owner of the ROCTITLE-ROC set to which the ROC will belong is obtained. The ROC is moved to the appropriate ROC field in working storage and is stored. If the error status indicates a duplicate, an error message is stored and there is a branch to the second paragraph of the loop. If the operation is successful, an accumulator for ROC records added is incremented, the record and its database key are displayed and there is a branch to the second paragraph of the loop. Once out of the loop, the two subscripts are set to zero. If an error occurred a routine to print error messages is performed. An accumulator for 'J1' add transactions is incremented and there is a branch back to the read paragraph.

The ROC delete routine is as follows: The UIC and level are used to locate the LEVEL record owner of the LEVEL-ROC set from which the ROCs are to be deleted. If it is not found, an error message is stored and there is a branch to an end of transaction routine. If the LEVEL-ROC set from which the ROCs are to be deleted is empty, an error message is stored and there is a branch to an end of transaction routine. The ROCs are deleted in a loop routine: A group subscript is incremented. If it exceeds 3, there is a branch out of the loop. A second subscript for ROCs within a group (on the card) is incremented. If it exceeds 4, it is reset to zero and there is a branch to the beginning of the loop. If the ROC field is blank, there is a branch to get out of the loop. The ROC record to be deleted is obtained using the ROC on the card. If it is not found, an error message is stored and there is a branch back to get another record. If it is found, it is erased, an accumulator for WSROC records deleted is incremented, and there is a branch to get another record.

Once out of the loop, the subscripts are set to zero. If any errors occurred, a routine to print error messages is performed. An accumulator for J1 delete transactions is incremented and there is a branch back to the read paragraph.

The end of transaction routine adds to an accumulator for J1 transactions with errors, performs a routine to print error messages, and branches back to the read paragraph.

The following paragraphs describe routines that are performed at various times during program processing. The routines' functions will be identified and the logic will be explained. The paragraphs will be referenced by their numeric portions only.

(1) 342 - 345 locate duplicate WS records and process them when appropriate. Prior to this routine, a WS record has been obtained. The logic follows: A duplicate WS record is obtained. If the error status indicates that there is no duplicate, there is a branch to get out of the routine. If the card being processed is an 'H1' or an 'H5' a routine is performed to determine if the WS is a member of a CLASS-WS set; otherwise a routine is performed to determine if the WS is a member of a SHIP-WS set. If the error status indicates no set membership in either case, there is a branch to the beginning of the routine. Otherwise, a true duplicate has been found and it is made to be the current record of run-unit. A duplicate counter is incremented. If the card type is 'G5' or 'H5', there is a branch to get out of the routine. Therefore, the following processes are performed only for 'G1' and 'H1' transactions. If the WS-WSROC set for the duplicate WS is empty there is a branch to the delete portion of the routine. Otherwise, the duplicate WS is made to be current record of run unit. If the card function is 'C' and the duplicate changed indicator is blank, a routine to modify the WS record is performed, the duplicate changed indicator is given a value of 'YES' and there is a branch back to the beginning of the routine. If the function is not 'C' or the duplicate indicator has a value other than spaces, the processing continues with a delete routine. The duplicate WS record is deleted, a routine is performed to obtain the WS whose database key is in hold, and there is a branch back to the beginning of the routine.

(2) 346 - 348 is used to obtain a WS record whose database key has been stored and then delete that record.

(3) 350 edits UIC : If the ship is not present on the card and in the SHIP file, an error message is stored.

(4) 355 - 356 is a SHIP record look-up: Given a UIC value the routine tries to obtain a SHIP in the SHIP file. If one is not found, an error message is stored.

(5) 360 edits class: If the class value is not numeric, an error message is stored.

(6) 365 - 366 is a CLASS record look-up: Given a class value the routine attempts to obtain a CLASS record. If one is not found, an error message is stored.

(7) 370 edits card function. If the function is neither 'A' nor 'B' nor 'C', an error message is stored.

(8) 380 edits watchstation: If the watchstation number is not numeric, an error message is stored.

(9) 390-395 is a WSTITLE look-up. Given a watchstation value, the routine attempts to locate a WSTITLE record. If one is not found, an error message is stored.

(10) 400 - 405 is a ROCTITLE look-up. Given a ROC value, the routine attempts to find a ROCTITLE record. If one is not found, an error message is stored.

(11) 407 locates a WS record via the SHIP-WS set: The routine obtains the next WS record of the SHIP-WS set. If there are no more (error status = '0307') an error message is stored and there is a branch to an exit paragraph. If the WS record found has the watchstation that is on the transaction being processed, there is a branch to an exit paragraph, otherwise there is a branch to the beginning of the routine.

(12) 408 locates a WS record via the CLASS-WS set: The routine obtains the next WS record of the CLASS-WS set. If there are no more, an error message is stored and there is a branch to an exit paragraph. If the correct WS is found, there is a branch to an exit paragraph; otherwise there is a branch to the beginning of the loop.

(13) 415 is an error routine: It is performed when an error status indicates a problem. The error status and card being processed are displayed along with an indicator to identify the paragraph where the problem occurred. Then there is a branch to the paragraph that prints edit totals.

(14) 420 - 435 print error messages stored when a transaction is being edited. The logic is as follows:

(15) If the line count exceeds 43, a routine to begin a new page is performed. The input transaction is moved to the print line and is printed. The card is underlined and another line is printed with spaces. The line count is incremented. There is a loop to print the error messages: A subscript (W-ERR-IND) is incremented by one. If the subscript is greater than the number of errors that occurred (w-CRD-ERRS), the loop is ended. There is a routine performed to determine whether to go to a new page and print headings, if necessary. An error message is moved from a table (W-PRT-ERR) to the print line and is printed. The line count is incremented. Then there is a branch to the beginning of the loop which continues until all error messages are printed. Once out of the loop, an accumulator for card type errors is incremented, two lines of spaces are printed, the table that held the error and the area that held the card in error are cleared, and the subscript and the error counter are re-initialized to zero.

(16) 435 is the routine to print report headers: The header routine sets the line count back to zero and adds 1 to the page counter. It then prints heading on a new page by moving to the print line headers that are set up in working storage and printing them one by one.

(17) 440 determines when to go to a new page: The line check routine checks to see if the line count is greater than 46, and if it is, the header routine is performed.

(18) 445 prints a line then re-initializes the print line to spaces.

(19) 450 - 465 print edit totals. The routine loops through a table where the totals are stored (W-CARD-EDT-TABLE). Before the loop begins, headers are printed on a new page, and counters for ROCs and WSROCs added and deleted are loaded into the table. The loop processing is as follows. A subscript (W-TYPE-IND) to identify the transaction or record type is incremented. If the subscript is greater than 6 (six groups of totals), the subscript is re-initialized to zero and there is a branch to an end of program routine. Another subscript (W-CTR-IND) is incremented (for the totals). If the subscript is greater than 5, it is re-initialized to zero and there is a branch to print the line. The data from the table is moved to the print line and printed. Then there is a branch to the beginning of the loop. The loop continues until all the edit totals are printed.

The last paragraph of the program closes all files and database areas and ends program processing.

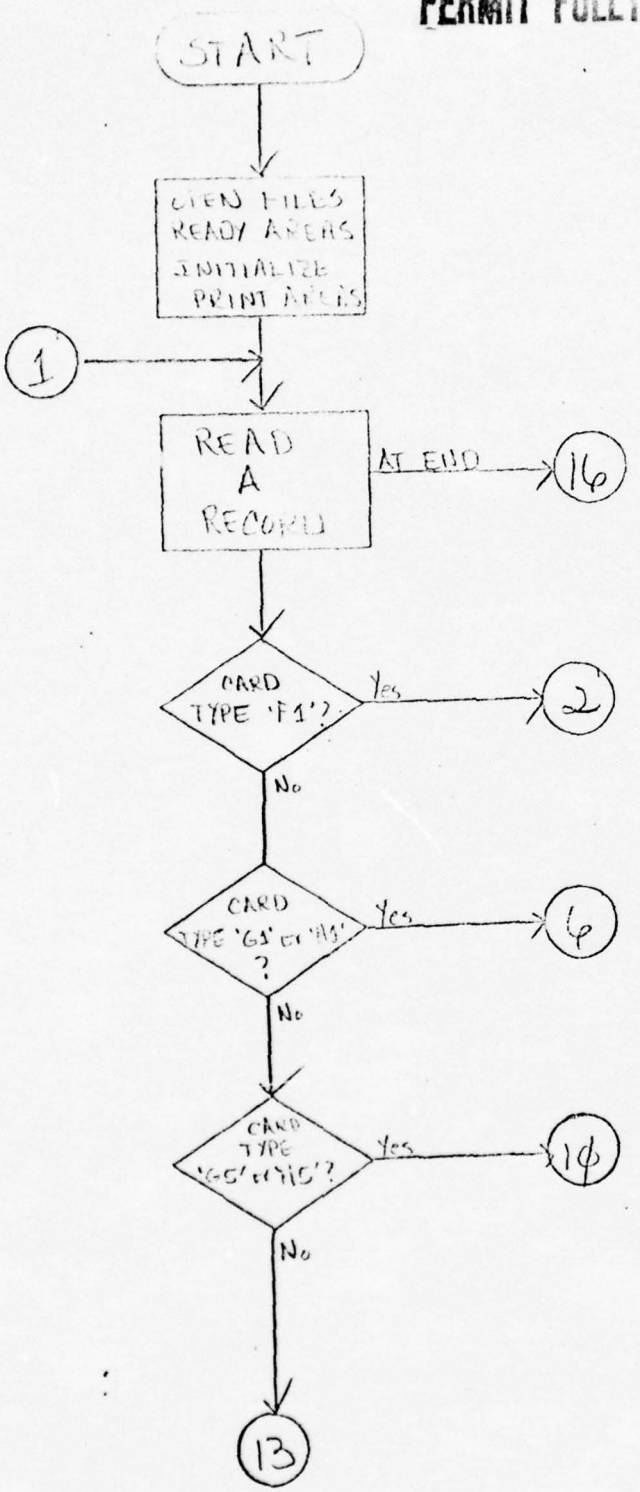
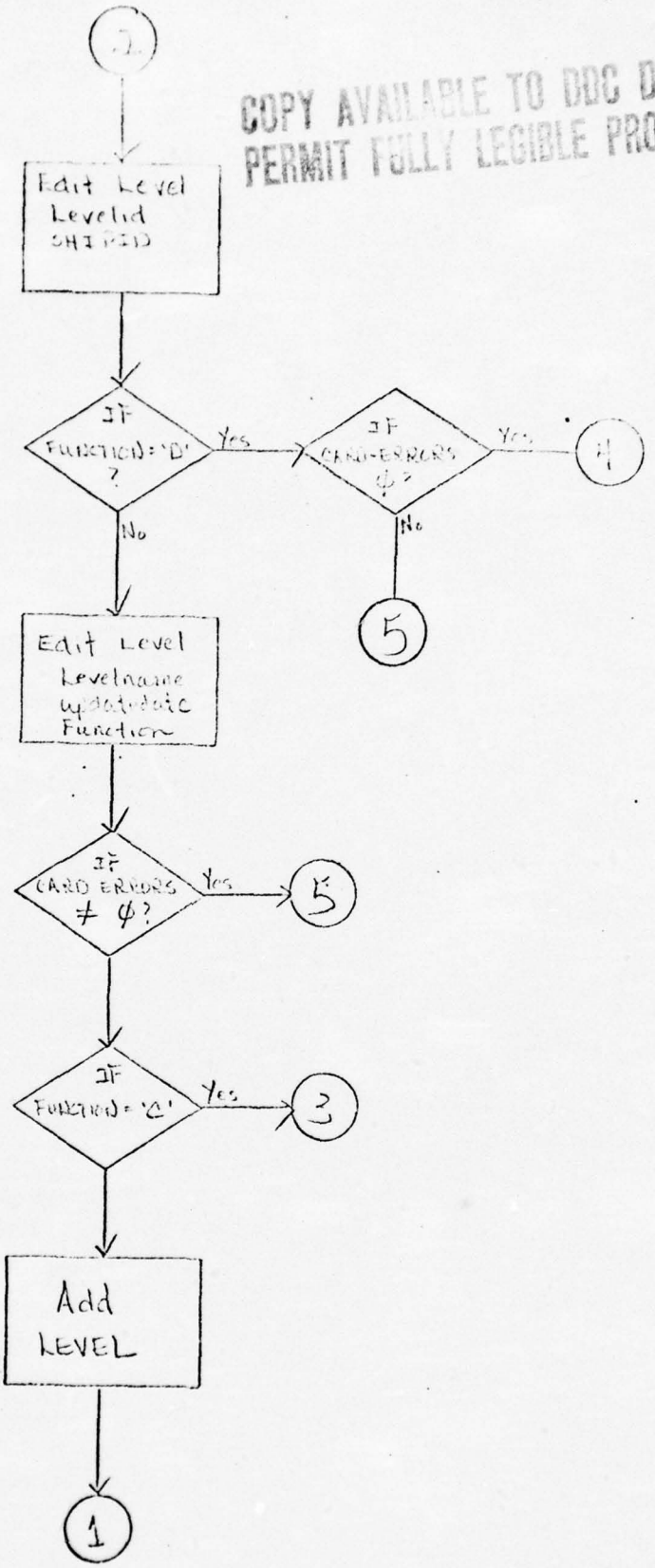
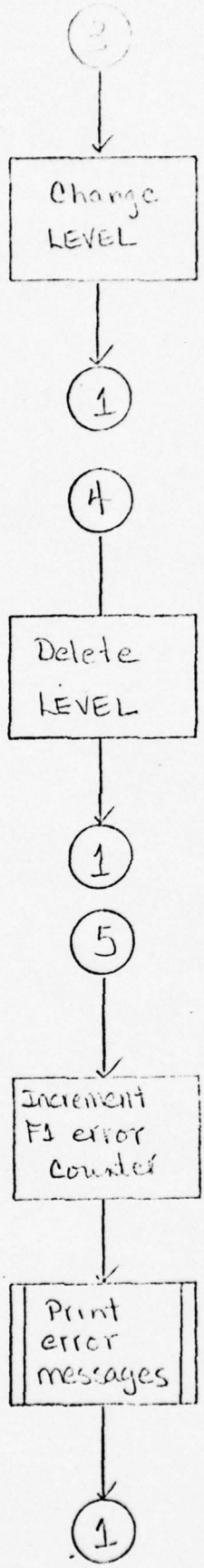
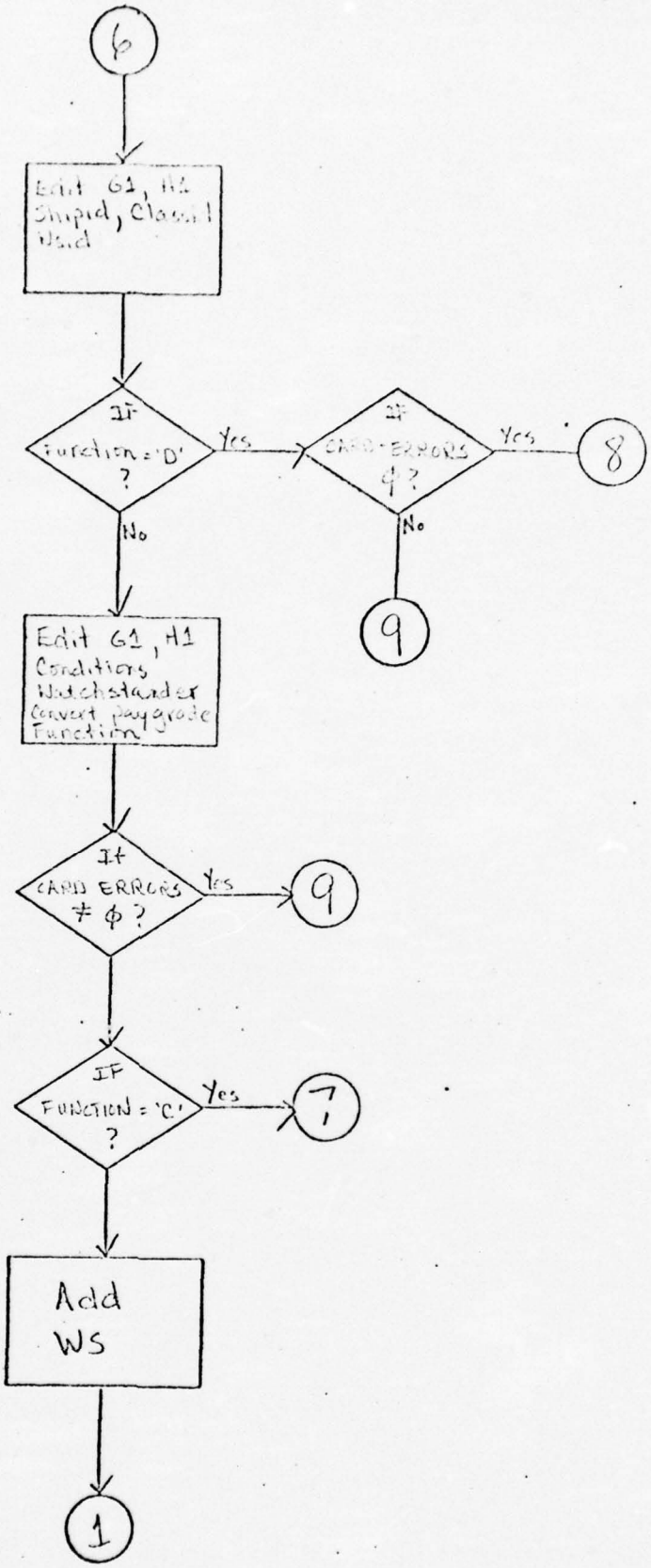
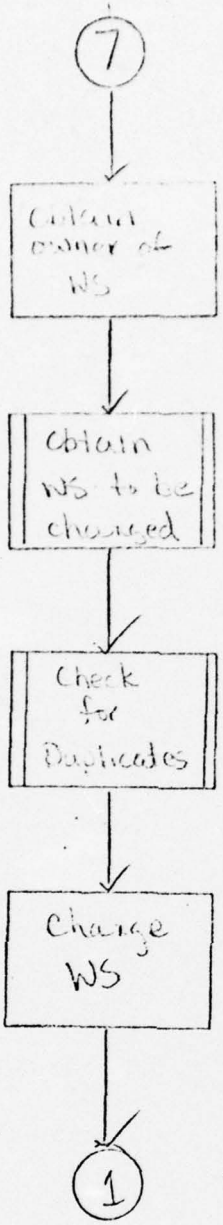


Figure E-1

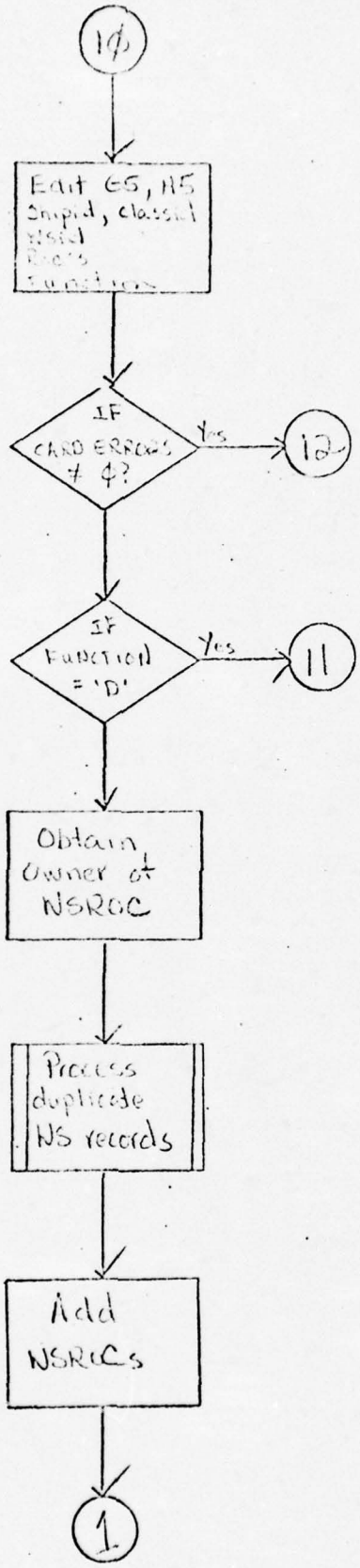


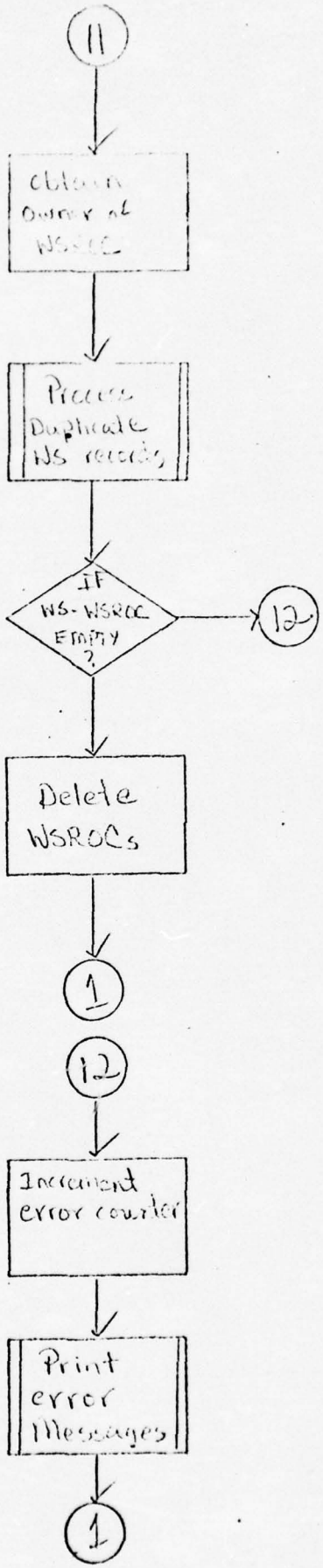

2
Change LEVEL
1
4
Delete LEVEL
1
5
Increment F1 error Counter
Print error messages
1

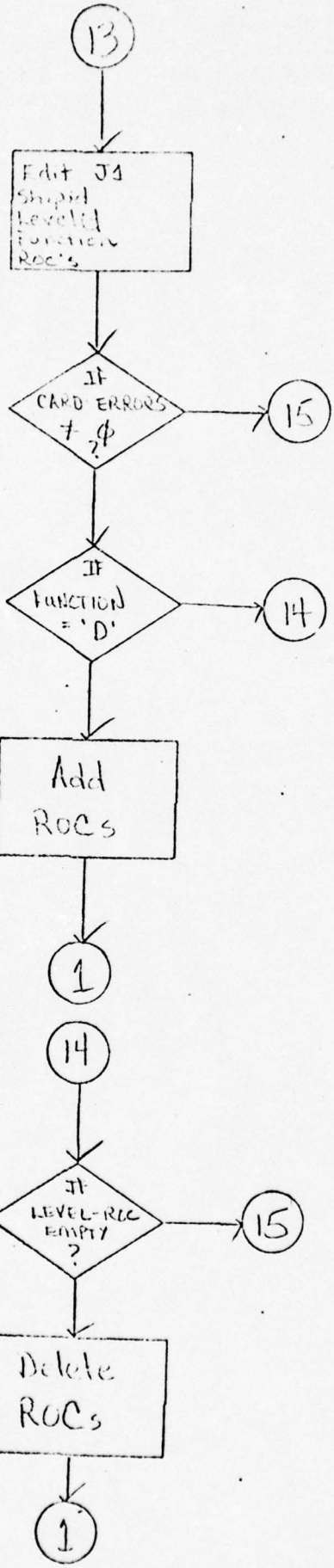

6
Edit G1, H1
Shipid, Classd
Used
IF
Function = 'D'
?
Yes
IF
CARD ERRORS
≠ ϕ ?
Yes
8
No
9
Edit G1, H1
Conditions
Watchstander
Convert paygrade
Function
IF
CARD ERRORS
≠ ϕ ?
Yes
9
IF
FUNCTION = 'C'
?
Yes
7
Add
WS
1

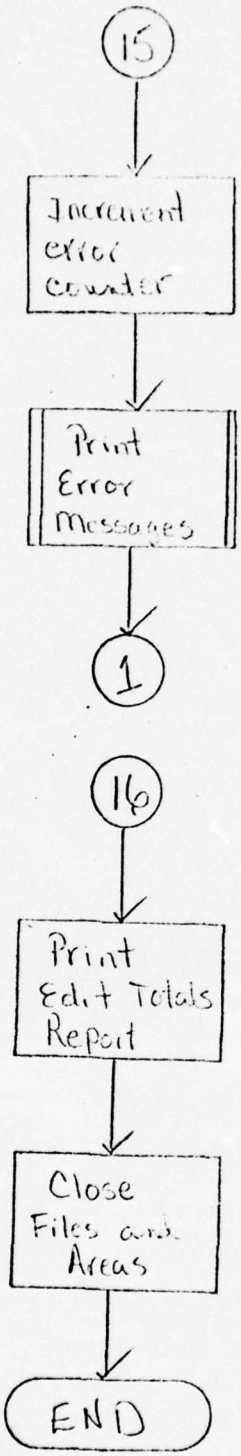

7
Obtain owner of WS
Obtain WS to be changed
Check for Duplicates
Change WS
1


8
Obtain Owner of WS
Obtain WS to be Deleted
Process Duplicate WS records
Delete WS record
1
9
Increment error Counter
Print error messages
1


10
Edit GS, NS
Shipid, classif
fixed
Rec's
Functions
IF
CARD ERRORS
≠ 4?
Yes
12
IF
FUNCTION
= 'D'
Yes
11
Obtain
Owner of
NSRec
Process
duplicate
NS records
Add
NSRecs
1


11
Obtain Owner of WS-REC
Process Duplicate WS records
IF WS-WSREC EMPTY ?
12
Delete WSRECs
1
12
Increment error counter
Print error Messages
1


13
Edit J1
Shaped
Leveled
Function
Roc's
If
CARD ERRORS
≠ ?φ
15
If
FUNCTION
= 'D'
14
Add
ROCs
1
14
If
LEVEL-ROC
EMPTY
?
15
Delete
ROCs
1


15
Increment error counter
Print Error Messages
1
16
Print Edit Totals Report
Close Files and Areas
END

APPENDIX F

MODULE NMWF01

Module Description. The Watch Station Generation Module, NMWF01, determines a minimum list of watchstations and watchstanders, by condition, for a specified ship and tasking level. The program was written under contract number N-00014-76-C-0473, effective 24 November 1975.

Module Functions. The following functions are performed by NMWF01:

For a given ship and tasking level, as specified by an NMRS Load or Produce command;

(1) Compute the minimum list of watchstations and watchstanders needed to satisfy the ship's Required Operational Capabilities Statement (ROC), and

(2) output that list as one watchstation record per watchstation, per watchstander, showing up to eight ship conditions with the respective watchstander requirements for each.

Accuracy and Validity. The Ship ROC/Watchstation Database is edited and validated in a series of other modules. The input stream (card images) is read to find NMRS Load (LOD) and Produce (PRD) Commands, which are further edited. For both card types, the following edits take place:

a. SEQNO (Columns 2-3) must equal '01',

b. TRANSCODE (Columns 6-8) must equal 'CMD',

c. UIC (Columns 12-16) must be found as the CALC key of a SHIP record on the database, and as the UIC part of the CALC key of a LEVEL record, and

d. TASKINGLEVEL (Columns 17-28) must be found as the LEVELID part of the CALC key of a LEVEL record.

For the Load command, the following additional edit takes place:

COMMAND (Columns 9-11) must equal 'LOD',

For the Produce command, the following additional edit takes place:

a. COMMAND (Columns 9-11) must equal 'PRD', and

b. RUNINDICATOR (Columns 29-30) must equal spaces.

Any input transaction card images which fail to meet any of these edit criteria bypass further processing. Only those transactions which satisfy all the above edits are used to produce a list of watchstations.

Inputs. Inputs to NMWF01 may be either or both of two different types of input transactions, plus six different Ship ROC/Watchstation Database record types. The acceptable input transaction types are:

a. NMRS Load Command, and

b. NMRS Produce Command.

The database record types used by NMWF01 are:

a. CLASS

b. LEVEL

c. ROC

d. SHIP

e. WS

f. WSROC

Outputs. Output from NMWF01 is the Watchstation List, DAXMSSWS.

Working Storage. Working Storage consists of various hold areas for data being reformatted; logical variables (switches); tables used for accumulating totals; accumulators for various totals; and subscripts used in accessing table elements.

Characteristics.

Input Stream Transactions.

a. IT-REC is the card input transaction.

b. Each input transaction is edited according to specifications for the particular record type associated with the transaction. These requirements are stated in '2.2 Module Functions'. If the record contains valid data, the transaction is used to generate a WATCHSTATION list for the SHIP/LEVELID; otherwise the transaction is bypassed.

c. Valid LOAD or PRODUCE Commands which specify a UIC and/or LEVELID not matching the CALC keys for the SHIP and LEVEL records cause appropriate diagnostic messages to be displayed, and are then bypassed.

d. All of the transaction data is static.

e. The input transaction file is a temporary disk file that requires one IBM-3330 disk track per 156 input records.

Database Records. The six database records used by this program are maintained by other programs based on input transactions subjected to a series of edits. Data edits include checks for non-blank data, numeric data, table lookups (e.g., MILTIPAY-GRADE, OEC), and lookups against Database records considered to comprise a dictionary of valid data (e.g., UIC on the SHIP record, CLASSID on the CLASS record, etc.). All data elements affecting the internal operation of the Ship ROC/Watchstation Subsystem are thoroughly edited to prevent errors.

However, a number of data elements which are merely passed through the Subsystem without change are not edited. This is because the edits are dependent on data in record types present on the NMRS Database, but not on the Ship ROC/Watchstation Database. Because of the inherent weakness of IDMS in not allowing a program to access more than one database at a time, it is not feasible to check these fileds at data entry time. Besides, as changes are made to the NMRS Database, it would be virtually impossible to revalidate the Ship ROC/Watchstation Database accordingly.

Therefore, it was decided that since all WATCHSTATION records input to DIP would necessarily be edited in any case (due to possible analyst input), then it would be logical to not attempt to edit them against an out-of-date copy of NMRS data at another time. Hence, such data elements as:

a. RATINGABBR

b. WSNEC

c. DESIGCODE

d. PAYPLAN

e. OCCUPATIONCODE

f. CIVPAYGRADE

g. ORGCODE

h. RESERVECODE

i. SEARCHORGKEY

would be edited only by numeric checks and non-blank checks, as appropriate, in the Ship ROC/Watchstation Subsystem. Each of these data elements, as well as the others, will be edited by DIP.

Further problems could be detected by DIP in its edit, especially in these data elements:

a. UIC

b. TASKINGLEVEL

c. WSID

d. CONDITION

due solely to changes to data on the NMRS Database which were not entered as corresponding changes to the Ship ROC/Watchstation Database. These would be errors of coordination, and not errors due either to the Ship ROC/Watchstation Subsystem or to DIP.

The only viable solution to such possible problems is to incorporate the Ship ROC/Watchstation Database into the NMRS Database, and to CALL and/or PERFORM all appropriate edit routines in DIP whenever the database is updated or WATCHSTATION lists are generated. This has already been discussed with the Database Administrator and his staff, and a number of steps have been taken to simplify this eventual conversion:

a. ANS COBOL/IDMS has been used as the source programming language.

b. Data element names have been coordinated (through DETS) with NMRS names.

c. Module documentation to NMRS standards has been provided for all Ship ROC/Watchstation modules.

d. NMRS programming conventions have been observed.

e. A database specification has been written according to the Database Administrator's specifications.

Output - the Watchstation File.

a. Identification.

(1) The DDNAME is DAXMSSWS.

(2) The DSNAME is WEUIDAX.MSS.SHIPROC.N.WS.

(3) The record name is OM-WATCHSTATION.

b. The Watchstation File is output to be the primary source for WATCH data in the NMRS Database.

c. The Watchstation File contains, for each specified ship and tasking level, a complete list of all watchstations required to be manned at any ship condition under the applicable Required Operational Capabilities Statement (ROC). Besides identifying the ship, tasking level, and watchstation, the record contains information on the minimum qualifications for the watchstander, e.g., skill-class, rating, rate, NEC, organizational component, etc. for each ship condition at which the watchstation is manned.

d. All of the watchstation data is static.

e. The Watchstation File is a disk file that requires one IBM-3330 disk track per 52 Watchstation records.

Data Environment.

The Ship ROC Table. For maximum efficiency of both main memory and CPU time, the Ship ROC table was developed to store the ROC data for the selected LEVEL. The table makes provision for ROC elements whose SOCID values are in the range 001 through 900. Each ROC record is loaded into the table by moving the letter 'X' (hexadecimal E7) to W-SOC-MARK(SOCID), where SOCID is used as the subscript to the table. The Ship ROC table is initialized with a period (hexadecimal 4B) in each cell. By subscripting the table with SOCID, random access against the table is achieved, rather than repeated, laborious sequential searches.

The Condition Table. The condition table contains an entry (the CONDITION data element) for each condition for which watchstations are to be manned.

Transaction Statitistics. The UIC; LEVELID, and count of ROC records within the LEVEL-ROC set are stored for up to 100 different ships and/or ROCs.

Program Logic. The Watchstation Generation program, NMWF01, is written in a hierarchical structure for ease of programming, testing, and maintaining. At its highest level is overall program control. This includes startup, processing, and endup routines. Included as startup functions are:

- Binding the IDMS run-unit and record types,
- Readying the IDMS realms,
- Opening the input transaction file and output watchstation file, and
- Initializing certain data elements.

Processing encompasses the bulk of data processing in this program, and will be discussed at length in subsequent paragraphs. Included as endup functions are:

- Finishing the IDMS database, and
- Closing the input transaction file and output watchstation file.

The program processing, apart from startup and endup, consists of processing transaction control cards, and processing watchstation data from the database. Each transaction card is read, and edited to determine if it meets the criteria detailed in Section 2.2.1 of this Specification. If a transaction card does not meet these criteria, it is bypassed, and the next transaction card is read, control returns to the topmost hierarchical routine, program control, to perform the endup process.

If a transaction card does meet these criteria, then the "ready for watchstation generation" flag is set. But before a watchstation list for the specified ship/tasking level is produced, a check is made against a table of previously processed ship/tasking levels to prevent generating a duplicate list. If the specified ship-tasking level has already been processed in the current run of the program, a message is displayed describing why another watchstation list is not being produced, and control is returned to continue reading transaction cards. Otherwise, processing begins for the specified ship/tasking level, as detailed in the following paragraphs.

The processing of a ship/tasking level is divided into three routines:

- Setup
- Process class watchstations, and
- Process ship watchstations.

The setup loads the ship's Required Operational Capabilities (ROC) into an in-core table. The process class watchstation routine walks the CLASS-WS set, processing all watchstation records common to all watchstations in the class of ships. The process ship watchstation routine walks the SHIP-WS set, processing all watchstations on the ship which are not common to the class.

The setup routine first obtains the LEVEL record, CALC on the LEVELKEY (SHIPID or UIC, and TASKINGLEVEL or LEVELID). If it is missing from the database, an error message is displayed, and control returns to processing the next transaction card.

If the LEVEL record is found, the LEVEL-ROC set is walked to load the in-core ship ROC table. The following steps occur until the ned of the LEVEL-ROC set is reached: the next ROC record in the LEVEL-ROC set is obtained; if the SOCID is numeric and less than 901, it is entered in the ship ROC table as an 'X' in the table element subscripted by SOCID; otherwise, an error message is dispalyed. When the end of the LEVEL-ROC set is reached, setup is completed, and control passes to process class watchstations.

To process class watchstations (WS records belonging to the CLASS-WS set), first the SHIP record is obtained, CALCed on UIC from the transaction card. If it is not on the database, an error message is displayed, and control passes back to process the next transaction card. Then the CLASS record is obtained as the owner in the CLASS-SHIP set. The CLASS record also owns a CLASS-WS set, which contains all records for all watchstations common to the entire ship class.

The CLASS-WS set is walked, checking each watchstation's driving WSROC elements to see if the watchstation is required by the ship's ROC, and this process continues until the end of the CLASS-WS set. First, the next WS record in the CLASS-WS set is obtained. If there are no WS records in the CLASS-WS set, display an error message. For each WS record, if its WS-WSROC set is empty, continue obtaining the next WS record until one is found whose WS-WSROC set is not empty.

When the owner of the WS-WSROC set is found, begin walking the WS-WSROC set, continuing until the watchstation is determined to be required by the ship's ROC, or the end of the WS-WSROC set is found. A watchstation is required if either a WSROC with SOCID = zero is found, or else a match is found between the SOCID and an entry in the ship ROC table. If the end of the WS-WSROC set is reached, then the current watchstation is not required under the ship's ROC for the given tasking level (LEVEL record). Control passes to continue obtaining the next WS in the CLASS-WS set. But if the watchstation is required, one or more watchstation records are generated from the WS records for the given watchstation.

Once a watchstation is determined to be required, obtain the first WS record with the current WSID CALC key. This allows all duplicate WS records to be obtained with the OBTAIN DUPLICATE statement later in the processing. Next, tabulate the ship conditions represented by all the duplicate WS records, up to a maximum of seven on any WS record, or a total of eight across all WS records. This is done for each WS record until either the CONDITION field is equal to spaces, or the seventh CONDITION has been processed from a single WS record. Then the next duplicate WS record is obtained, and processed similarly. In each case, the CONDITION is entered into the condition table. If NOWATCH-STANDERS on the WS record is numeric, it is entered in the condition table; otherwise, 1 is entered as the default NOWATCHSTANDERS.

Again, obtain the first duplicate WS record, by CALC. Now, load the WATCHSTATION record by moving data from WS records to the corresponding WATCHSTATION record data elements, filling up to eight conditions on the WATCHSTATION record. If NOWATCHSTANDERS is greater than 1 for one or more conditions, additional WATCHSTATION records will be produced for those conditions, decrementing the number of additional watchstanders on the condition table by 1 each time a new one is written.

As each WATCHSTATION record is fully loaded, move it to a hold area, and write it to the WATCHSTATION file, DAXMSSWS. Then move from the hold area to the new buffer for use in writing the additional WATCHSTATION records due to more than one watchstander under one or more conditions. When all WATCHSTATION records are written for the given WSKEY, return control to process the next WS record in the CLASS-WS set.

Finally, when the CLASS-WS set end is reached, process the SHIP-WS set. Obtain the SHIP record, walk to SHIP-WS set, and process exactly the same as for the CLASS-WS set. At the end of the SHIP-WS set, all watchstations for the ship/tasking level have been processed. Enter the ship/tasking level in a table of run statistics to prevent the run from being duplicated by a subsequent LOAD or PRODUCE command. Then return control to process the next transaction card.

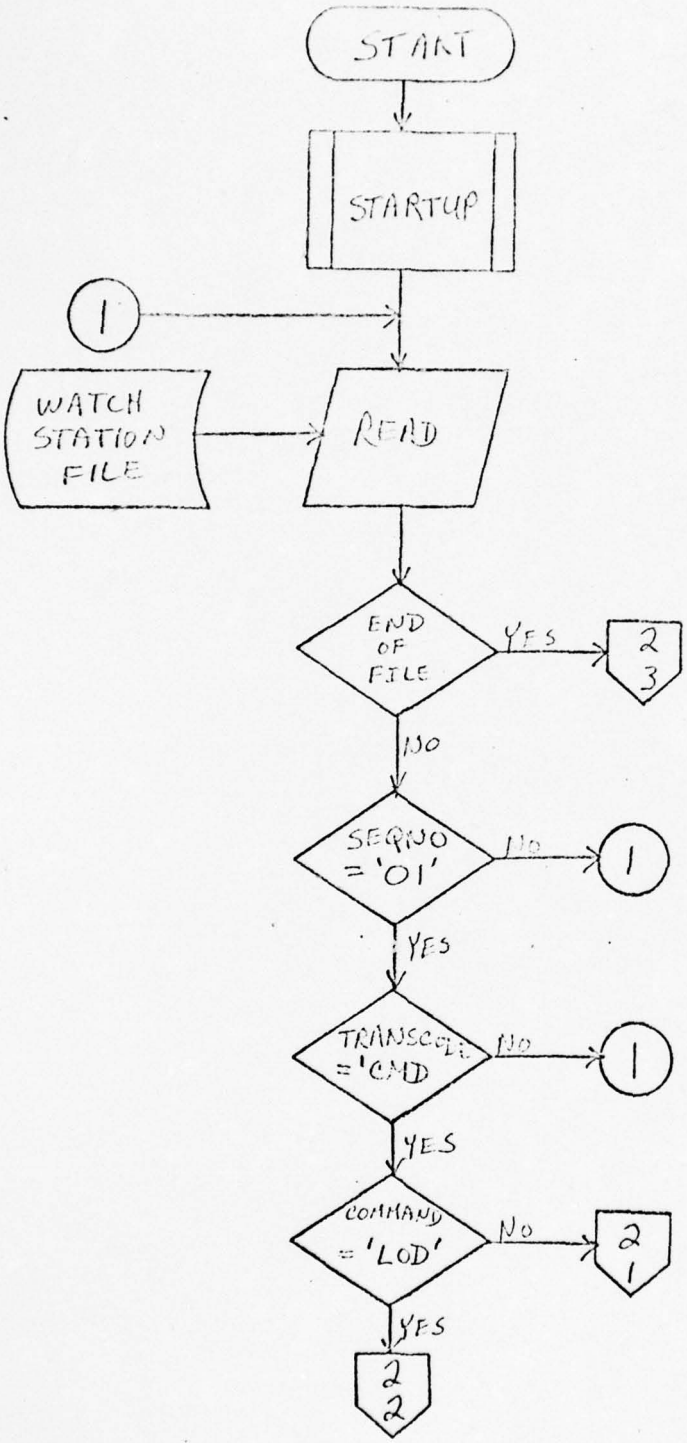
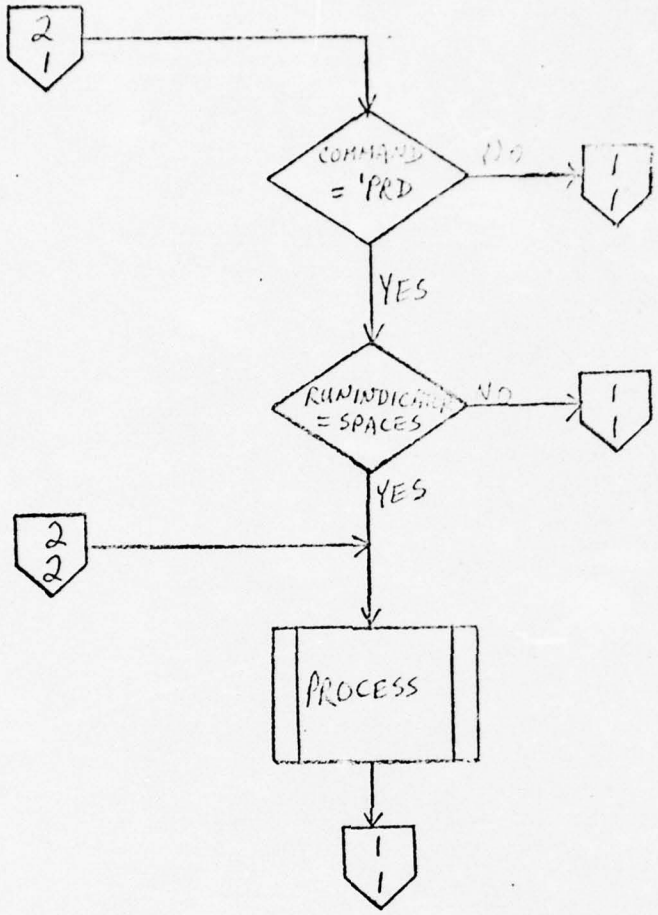
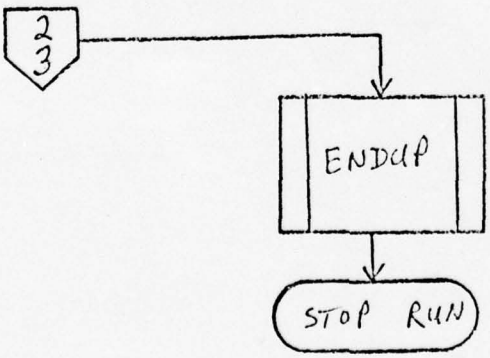
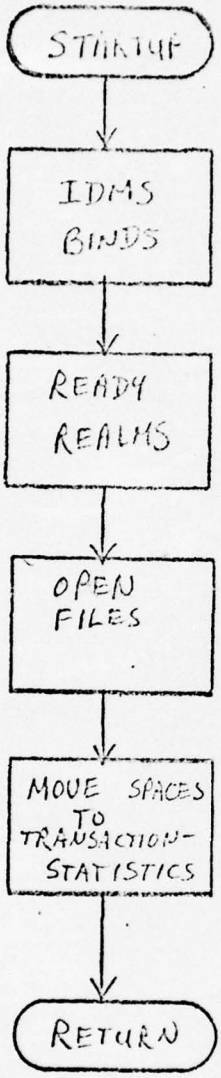
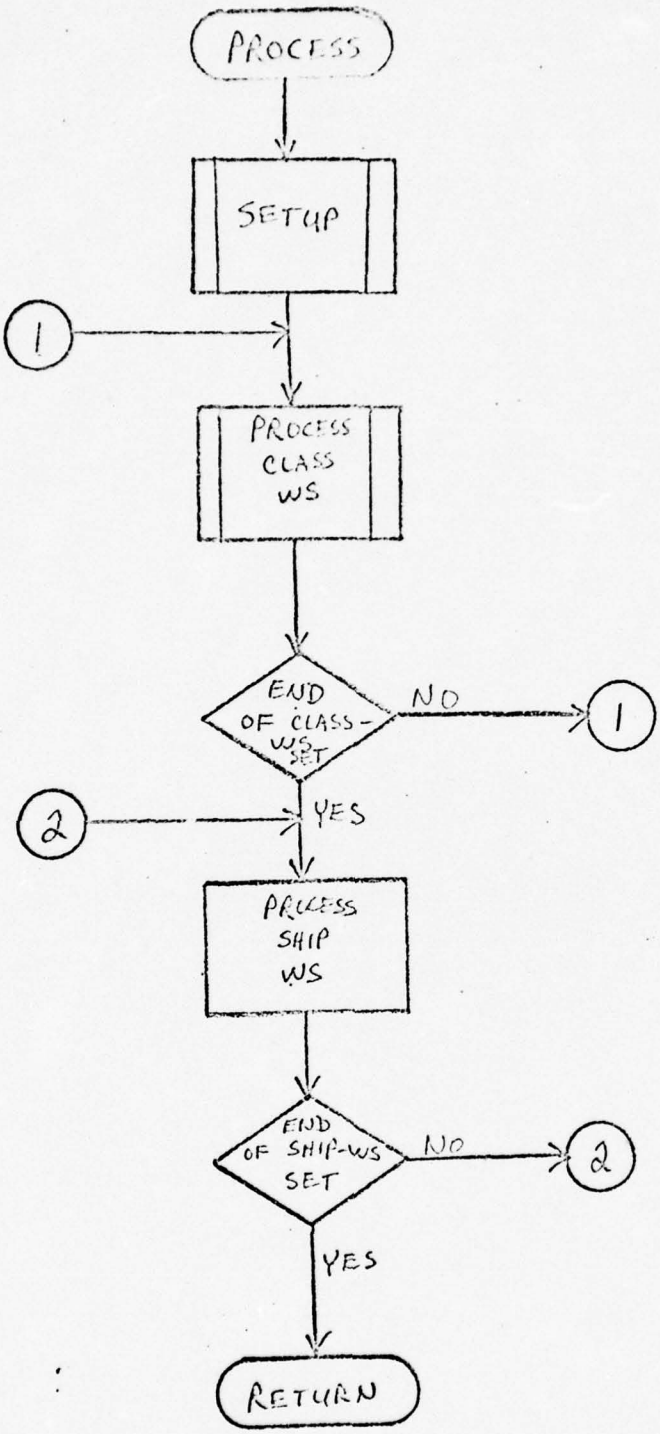


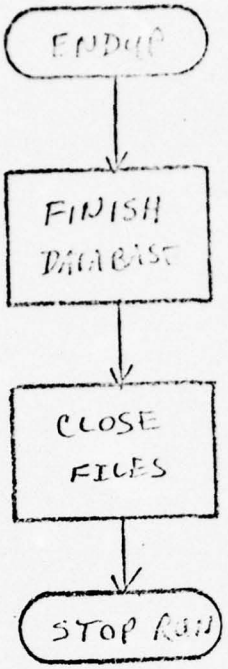
Figure F-1 NMWF01 Module Flowchart.

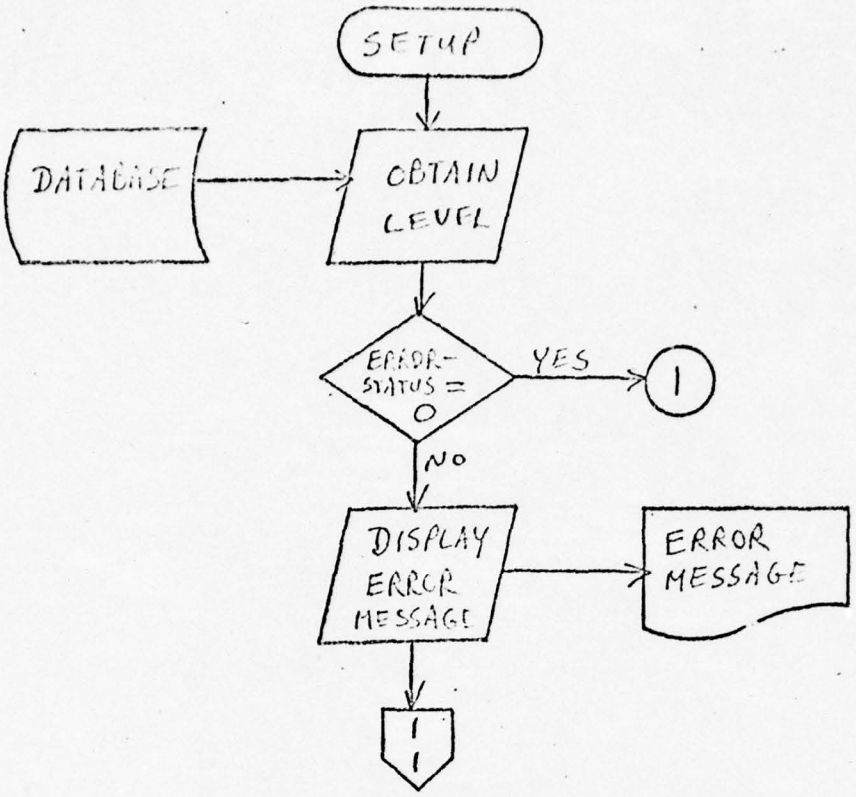

2
1
COMMAND
= 'PRD
NO
1
1
YES
RUNINDICATOR
= SPACES
NO
1
1
YES
2
2
PROCESS
1
1

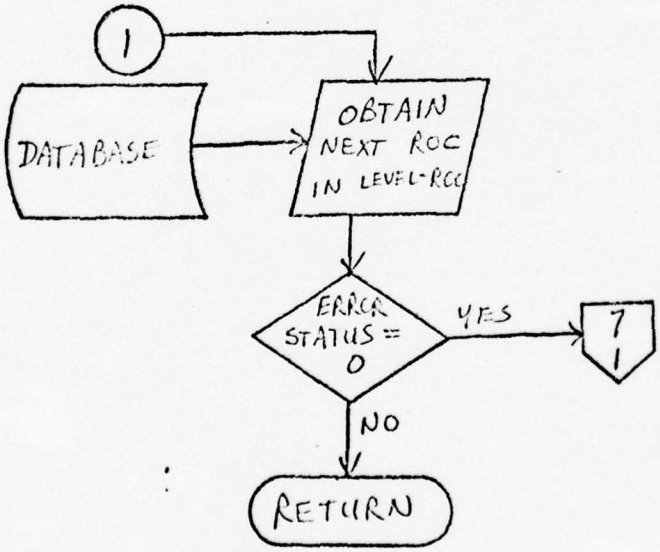

2
3
ENDUP
STOP RUN

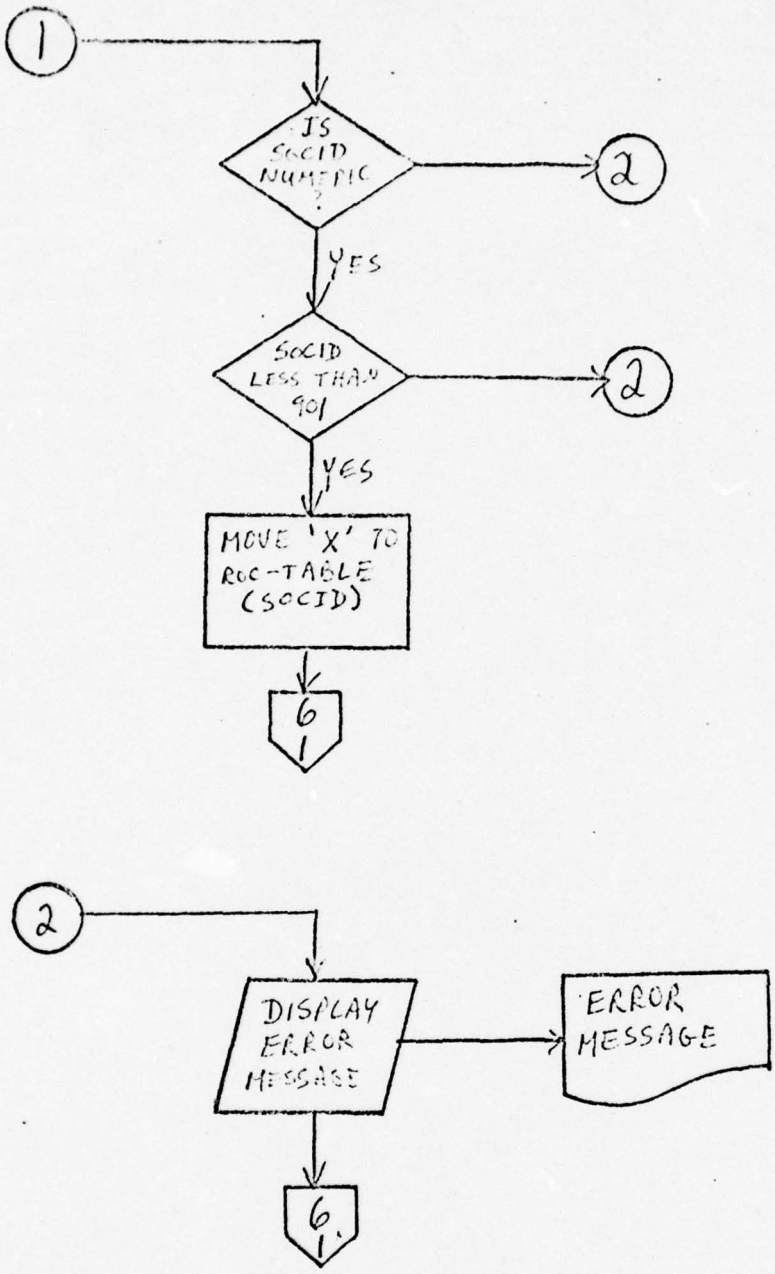

STARTUP
IDMS BINDS
READY REALMS
OPEN FILES
MOVE SPACES TO TRANSACTION-STATISTICS
RETURN

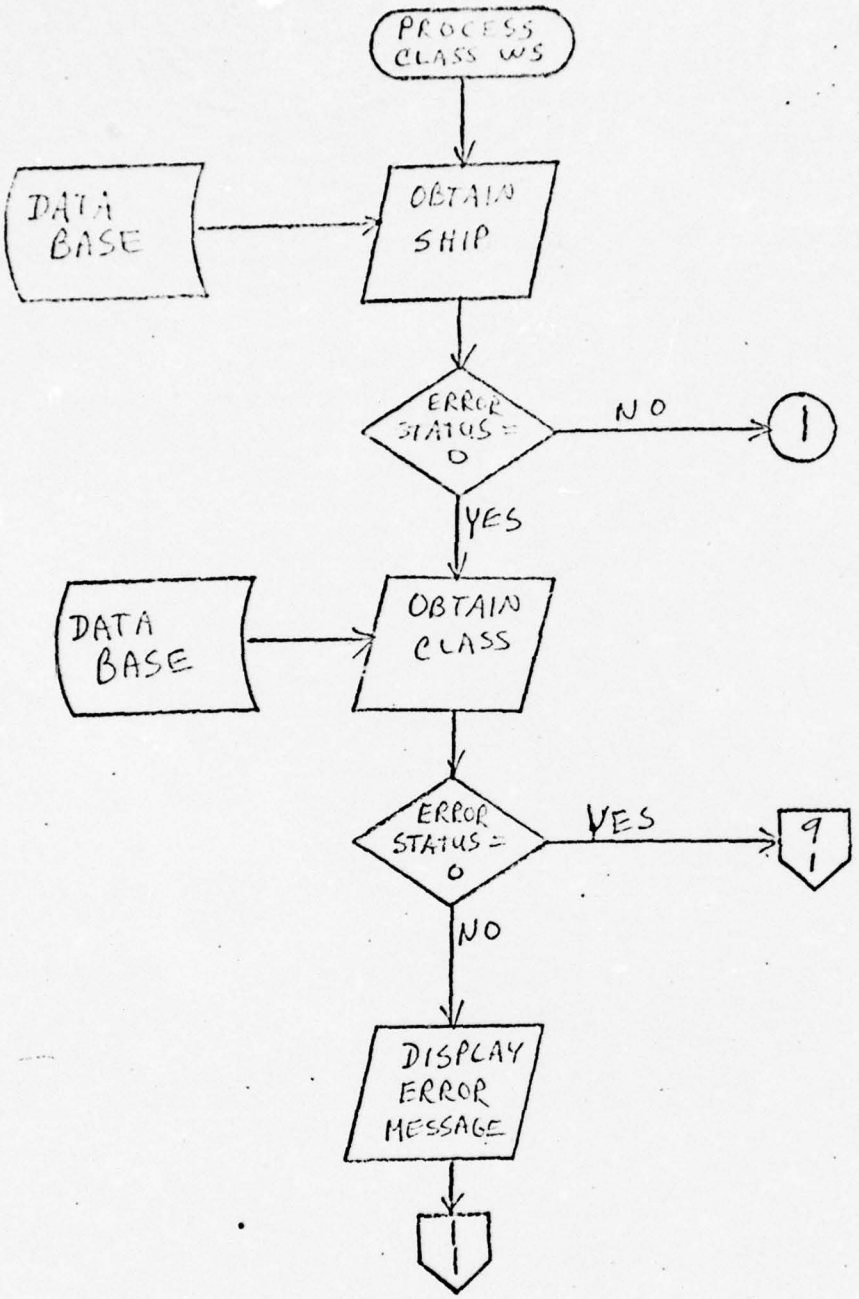

PROCESS
SETUP
1
PROCESS CLASS WS
END OF CLASS-WS SET
NO
1
2
YES
PROCESS SHIP WS
END OF SHIP-WS SET
NO
2
YES
RETURN

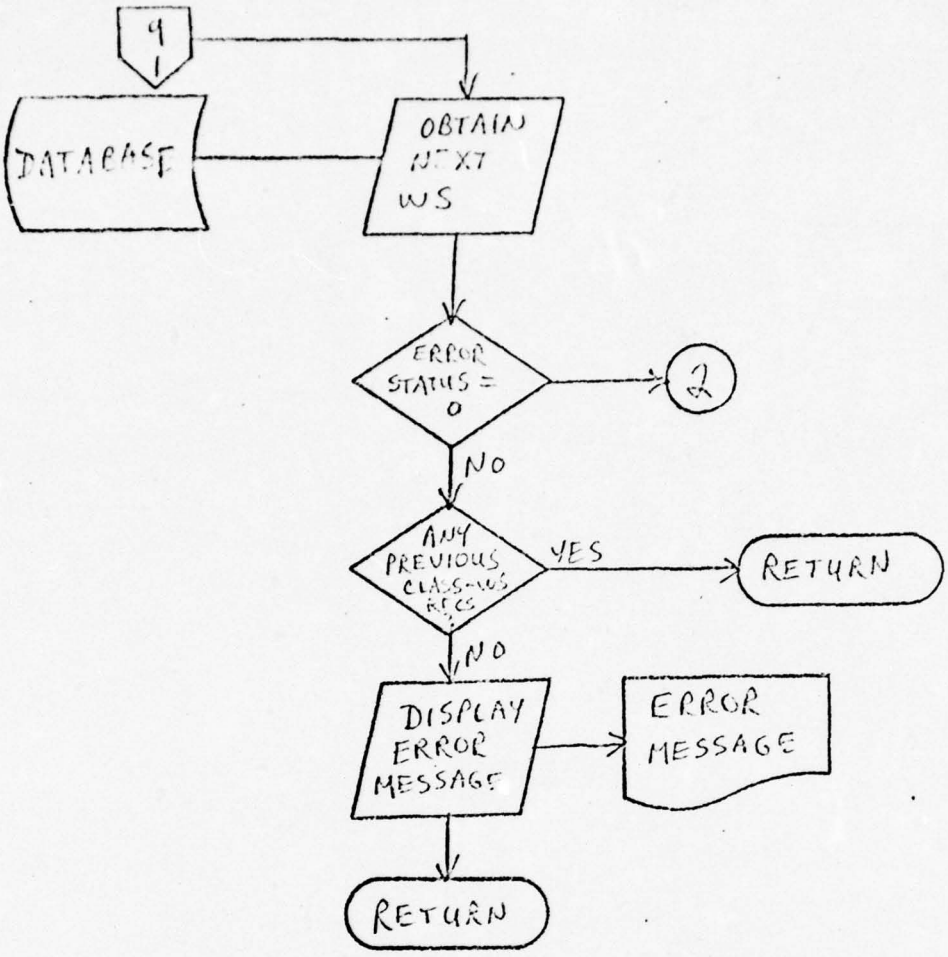

ENDOF
FINISH
DATABASE
CLOSE
FILES
STOP RUN

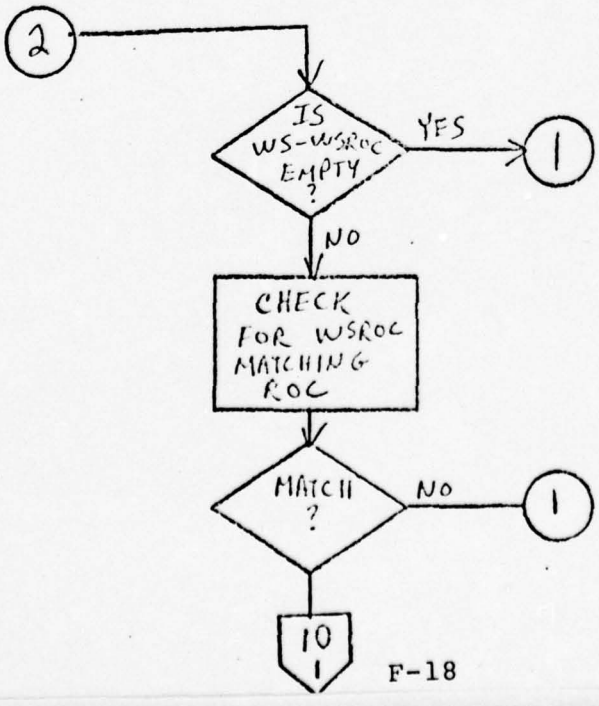

SETUP
DATABASE
OBTAIN LEVEL
ERROR-STATUS = 0
YES
1
NO
DISPLAY ERROR MESSAGE
ERROR MESSAGE
1

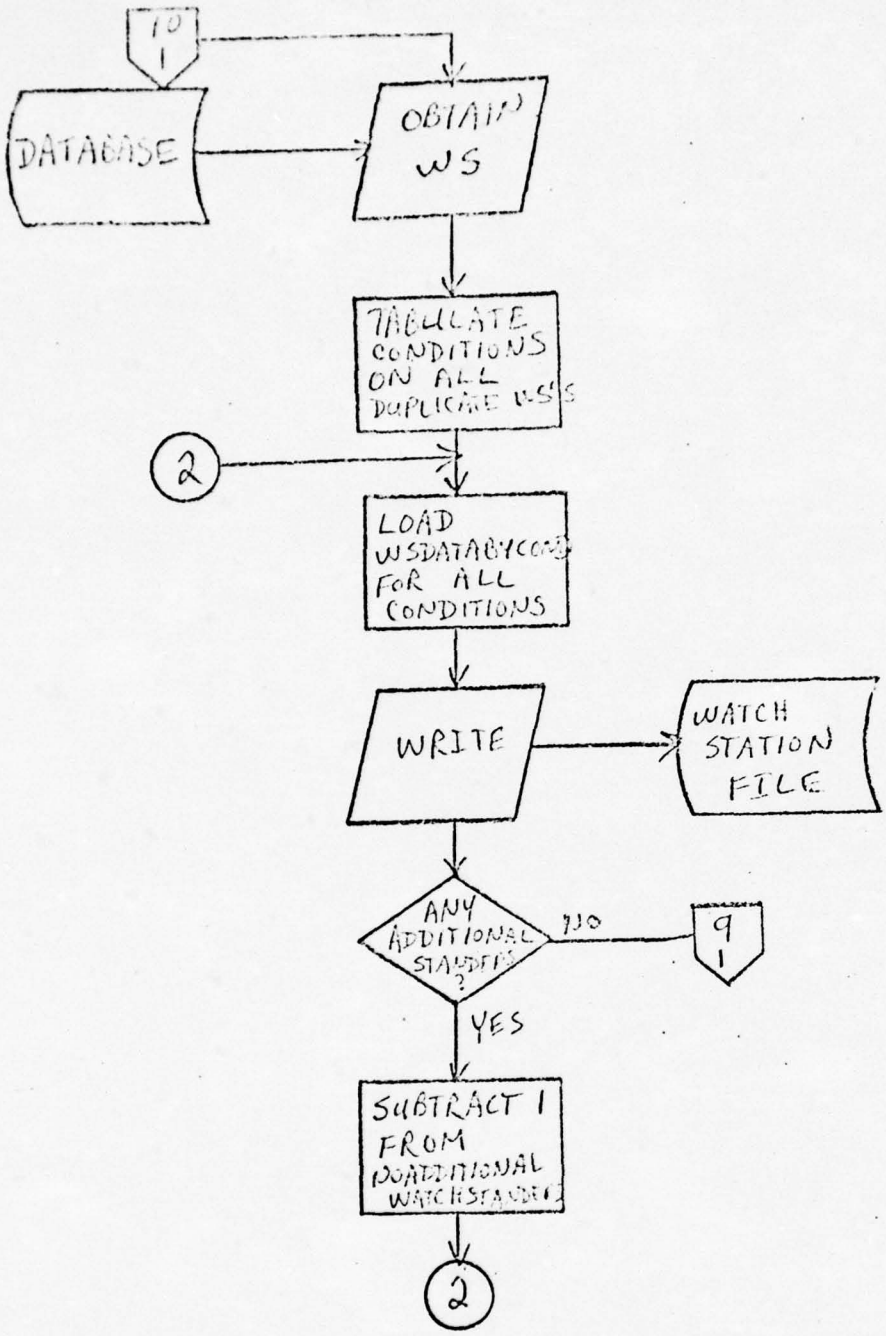

1
DATABASE
OBTAIN NEXT ROC IN LEVEL-ROC
ERROR STATUS = 0
YES
7
NO
RETURN

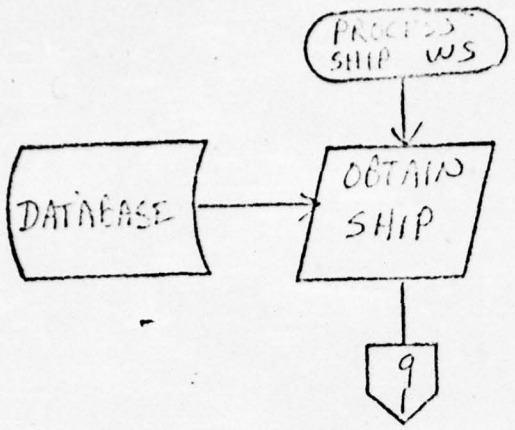

1
IS SOCID NUMERIC ?
2
YES
SOCID LESS THAN 901
2
YES
MOVE 'X' TO ROC-TABLE (SOCID)
6
1
2
DISPLAY ERROR MESSAGE
ERROR MESSAGE
6
1


PROCESS CLASS WS
DATA BASE
OBTAIN SHIP
ERROR STATUS = 0
NO
1
YES
DATA BASE
OBTAIN CLASS
ERROR STATUS = 0
YES
9
1
NO
DISPLAY ERROR MESSAGE
1


9
1
DATABASE
OBTAIN NEXT WS
ERROR STATUS = 0
2
NO
ANY PREVIOUS CLASS-WS RECS ?
YES
RETURN
NO
DISPLAY ERROR MESSAGE
ERROR MESSAGE
RETURN


2
IS WS-WSROC EMPTY ?
YES
1
NO
CHECK FOR WSROC MATCHING ROC
MATCH ?
NO
1
10
1


10
1
DATABASE
OBTAIN WS
TABULATE CONDITIONS ON ALL DUPLICATE WS'S
2
LOAD WSDATABYCOND FOR ALL CONDITIONS
WRITE
WATCH STATION FILE
ANY ADDITIONAL STATIONS?
NO
9
1
YES
SUBTRACT 1 FROM NOADDITIONAL WATCHSTATIONS
2


PROCESS SHIP WS
DATABASE
OBTAIN SHIP
9

APPENDIX G

MODULE NMWF02

Module Description. The Watchstation Conversion Module converts the physical sequential Watchstation File to the Indexed Sequential Access Method, ISAM, for input to DIP. The program was written under Contract Number N-00014-76-C-0473, effective 24 November 1975.

Module Functions. The following functions are performed by NMWF02:

For a given ship and tasking level:

(1) Read each Watchstation record, one at a time.

(2) Assign a 4-digit sequence number to each record, beginning with '0001' and incrementing by 1.

(3) Write each Watchstation record as an ISAM record whose key is UIC, Tasking level, and sequence number.

Inputs. The input to NMWF02 is the Watchstation File, DAXMSSWS, created in program NMWF01.

Outputs. Output from NMWF02 is the Watchstation List as an Index Sequential (ISAM) file.

Characteristics.

Input - the Watchstation File.

a. Identification.

(1) The DDNAME is DAXMSSWS.

(2) The DSNAME is WEUIDAX.MSS.SHIPROC.N.WS.

(3) The record name is IM-WATCHSTATION.

b. The Watchstation File is output from NMWF01 to be the primary source for WATCH data in the NMRS Database, and input to NMWF02 for reformatting as an ISAM file for easier retrieval by DIP.

c. The Watchstation File contains, for each specified ship and tasking level, a complete list of all watchstations required to be manned at any ship condition under the applicable Required Operational Capabilities Statement (ROC). Besides identifying the ship, tasking level, and watchstation, the record contains information on the class, rating, rate, NEC, organizational component, etc. for each ship condition at which the watchstation is manned.

d. All of the watchstation data is static.

e. The Watchstation File is a disk file that requires one IBM-3330 disk track per 52 Watchstation records.

Output - the ISAM Watchstation File.

a. Identification.

(1) The DDNAME is DAXMSSDI.

(2) The DSNAME is WEUIDAX.MSS.SHIPROC.N.DI.

(3) The record name is OM-WATCHSTATION.

b. The ISAM Watchstation file is output to be the primary source of WATCH data in the NMRS database.

c. The file contents are identical to those for DAXMSSWS, above.

d. All of the watchstation data is static.

e. The ISAM Watchstation File is an Indexed Sequential (ISAM) disk file that requries space in cylinders. One track per cylinder is designated for the system index and overflow area. The remaining 18 tracks per cylinder each will hold up to 44 watchstation records.

Data Environment.

Job Statistics. A count of records written, and a computation of IBM-3330 disk drive tracks and cylinders needed to hold the ISAM watchstation file is presented for use in tracking the utilization of disk storage, and for modifying the JCL accordingly.

Logical Variables. A logical variable is used to signal the end of the input watchstation file, and to trigger program termination procedures.

ISAM Key. The ISAM key for the output Watchstation file is built here, from the UIC and LEVELID of the input Watchstation File and a 4-digit sequence number.

Program Logic. The Watchstation to ISAM program, NMWF02, reads the Watchstation file produced by NMWF01 after is has been sorted as follows:

a. UIC (ascending),

b. LEVELID (ascending), and

c. WSID (ascending).

First, it opens the input Watchstation File and the output ISAM file. Then it processes watchstation records until end-of-file is reached on input. Finally, it computes certain job statistics, displays them, closes the files, and terminates.

The processing of watchstation records consists in reading them from the Watchstation File, building an ISAM key, and writing them to the ISAM file. The ISAM file key consists of the UIC and LEVELID from input, plus a sequence number. The sequence number is reset to '0001' whenever either UIC or LEVELID changes on input, and is incremented by 1 thereafter. Records are counted as they are written.

The job statistics consist of records written (as counted above) and the IBM-3330 tracks and cylinders needed to store the data. There are up to 44 records per track, and 18 data tracks per cylinder (as the nineteenth track is reserved for the index and the overflow area). These statistics are computed and displayed, after which the files are closed and the run terminated.

Program Flowchart NMWF02

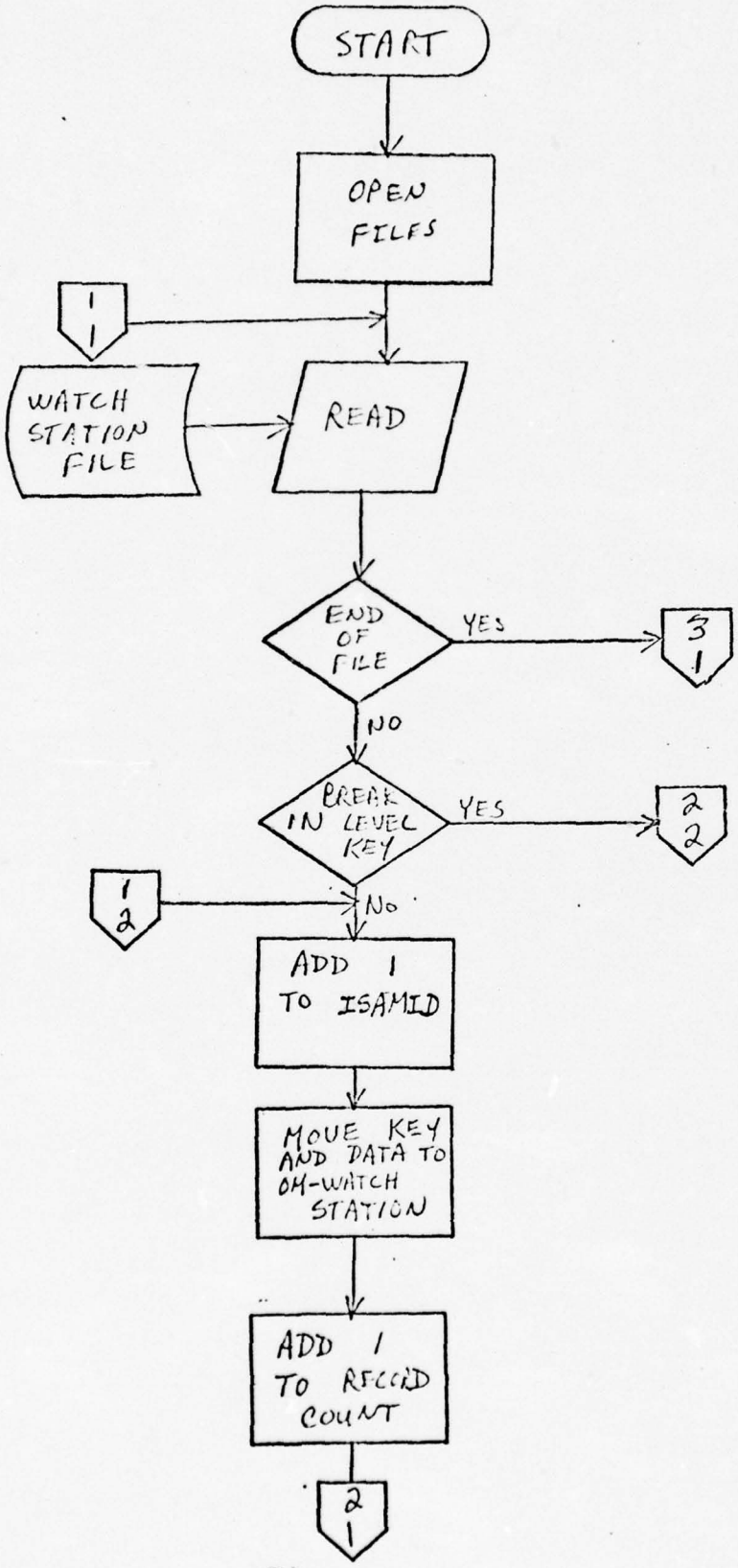
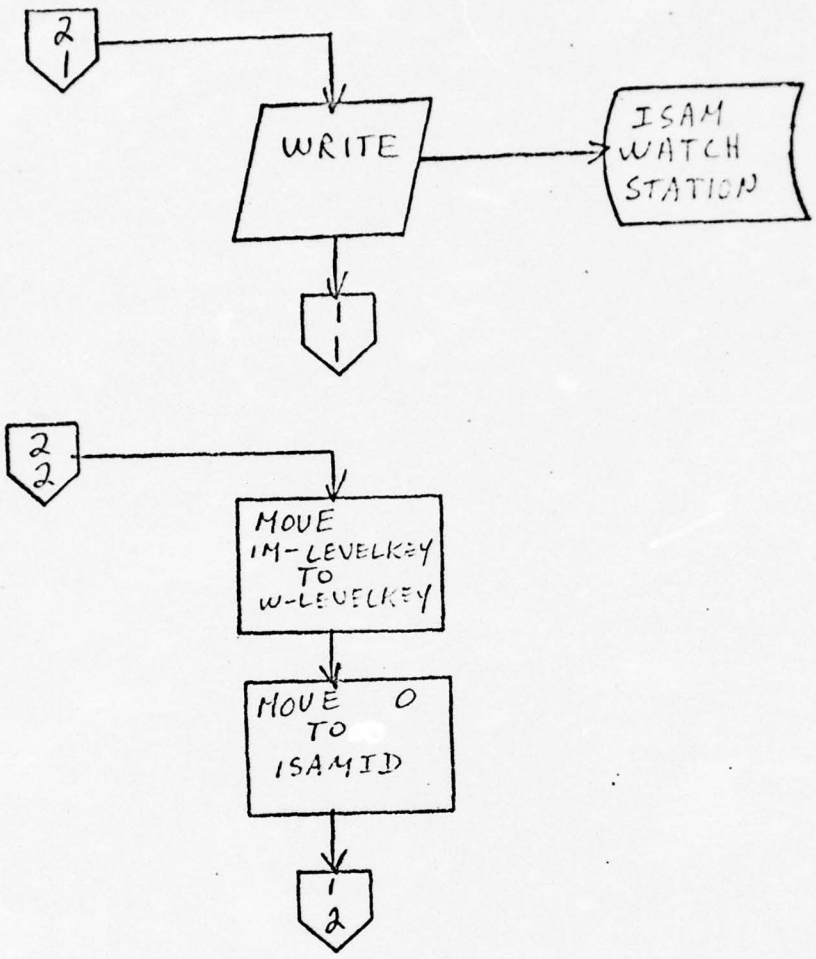
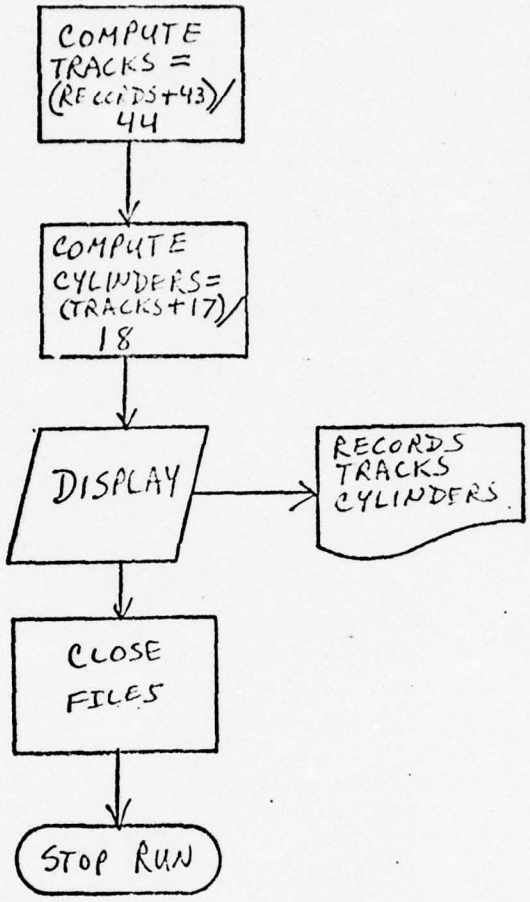


Figure G-1


2
1
WRITE
ISAM
WATCH
STATION
1
1
2
2
MOVE
IM-LEVELKEY
TO
W-LEVELKEY
MOVE 0
TO
ISAMID
1
2


COMPUTE TRACKS = (RECORDS+43)/ 44
COMPUTE CYLINDERS= (TRACKS+17)/ 18
DISPLAY
RECORDS
TRACKS
CYLINDERS
CLOSE FILES
STOP RUN

APPENDIX H

DATA BASE

Data Base Development Outline

1. Conceptual Design

   a. Identify major data entities

      (1) Define each data entity
      (2) Identify source
      (3) Characterize usage

   b. Identify inter-relationships of data entities

      (1) Define each relationship
      (2) Identify the data redundancies

   c. Design data structure

      (1) Constructing pro-forma data structure
      (2) Test for logical completeness
      (3) Document data structure

2. Data Base Design

   a. Identify record types

      (1) Group data elements by data entities
      (2) Define each record type
      (3) Characterize usage:

         (a) Subsystems to access this record
         (b) Unique identifier
         (c) Secondary key
         (d) Volume estimate

   b. Identify inter-relationships of records

      (1) Identify set-types

         (a) Define owner
         (b) Define member(s)

      (2) Define each set

         (a) Description
         (b) Purpose

   c. Design data structure

      (1) Combine records/sets into a data structure
      (2) Identify primary access paths
      (3) Identify major subsystem usage of the data structure (SCHEMA)
      (4) Design data structure (Sub-SCHEMA) for major subsystems

d. Formalize data base design

(1) Document the data structure (SCHEMA)
(2) Document the data structure (Sub-SCHEMA)

CONCEPTUAL DESIGN

This section of the Data Base Development specification is written to provide the information necessary to define the major data entities. It identifies the source of each data entity within the structure. In addition, it will provide a variety of background information and discussion relative to the derivation of the data structure diagram depicted in Figure H-2.

Identification of Major Data Entities

This paragraph includes a definition of each data entity and identifies the source of each data entity.

A. CLASS

The Ship Class record identifies a class of ships with reference to the prototype ship of the class. The class is identified by a three digit number, CLASSID, which was arbitrarily assigned. It contains pointers to ship and watchstation records.

Data Source - The original dictionary of approximately 110 ship classes was compiled by the Ship ROC/Watchstation team in consultation with NAVMMACLANT personnel. Additional ship classes are to be assigned CLASSID's in-house at NAVMMACLANT as needed.

B. CONDTITLE

The Condition Title record relates the two character ship CONDITION data element (e.g., 1) to the Condition Title (e.g., CONDITION I).

Data Source - The original Condition Title dictionary contains 11 ship conditions and their titles, and was compiled by Ship ROC/Watchstation team in consultation with NAVMMACLANT personnel. Additional condition titles are to be assigned CONDITIONs in-house at NAVMMACLANT as needed.

C. LEVEL

The level record aggregates the elements of a specific ship's Required Operational Capabilities Statement (ROC). A ship may have an assigned ROC, and many hypothetical ROCs under consideration for their manpower impact. Each of these ROCs is given a unique LEVEL record, identified by its LEVELID for a given ship, to identify the ROC and group the ROC elements together. It is important to note that the ROC records attached to a LEVEL record take on a classification of CONFIDENTIAL in that an actual ship's ROC is confidential.

Data Source - The level record is created by an analyst whenever he wants to create a new (actual or hypothetical) ROC for a ship. The LEVEL record must be created before any ROC records (ROC elements of the tasking level) are entered to the system.

D. ROC

The ROC record identifies a single ship's operational or suboperational capabilities of a ship's Required Operational Capabilities Statement(ROC). The combination of all ROC records within a tasking level constitutes a ship's ROC. The ROC element is encoded as a 4-digit number which can be decoded by reference to the ROC title record. It is important to note that the ROC records are classified CONFIDENTIAL when they are associated with a tasking level or ship.

Data Source - ROC records are entered as needed by the analyst within a tasking level to completely specify a (real or hypothetical) ship's ROC.

E. ROCTITLE

The ROC Title record relates the 4-digit code for a single ship's operational or suboperational capability (ROC element) to the actual Navy 8-character code for a ROC element. The Navy code is of the form MMM ii.jj, where MMM is the mission area (MOBility, Command And Control, etc.), ii is the operational capability number, and jj identifies the suboperational capability.

Data Source - The ROC Title dictionary contains all operational and suboperational capabilities identified in OPNAVINST 3501.D, identified by a 4-digit code and optionally by the 8-character official Navy code. A 4-digit code was arbitrarily assigned to each of the official codes by the Ship ROC/Watchstation team to simplify data gathering, reduce the keypunching, disk access, and disk storage costs for ROC elements by 50%, and provide a possible technique for encoding ship ROCs (by not identifying the 8-character official Navy code for them on this record type) so as to meet the security requirements on a non-secure computer facility. Additional ROC Titles are to be assigned 4-digit codes in-house at NAVMMACLANT.

F. SHIP

The Ship record identifies a ship by its hull number, UIC, and name, and points to the ship class of which it is a member.

Data Source - The original dictionary of approximately 500 ships was compiled by the Ship ROC/Watchstation team in consultation with NAVMMACLANT personnel. Updates to the ship dictionary are to be made in-house at NAVMMACLANT as needed.

G. WS

The Watchstation record identifies the Watchstation by its 10-character ID number, and by the ship class or individual ship with which it is associated. It identifies the watchstander by RATE, RATING, and NEC (if enlisted) or Officer Designator, Pay Grade, and NOBC (if an officer). Further, it specifies the ship conditions under which the given watchstander mans the watchstation, the organizational component from which he comes, and whether or not the watchstation can be manned around the clock by two people rather than three (the SURGE condition).

Data Source - The Ship ROC/Watchstation team has coded all watchstation data for 68 ships as of 23 April 1976. The data was derived from approved SMD II documents where available, and the best available data otherwise, in consultation with NAVMMACLANT personnel. Watchstation data for the remaining ships is to be gathered, coded, and entered to the data base using NAVMMACLANT in-house or contractor personnel as directed.

H. WSROC

The Watchstation/ROC record identifies an operational or sub-operational capability which, to be properly performed, requires the manning of the specified watchstation. Each watchstation must have one or more WSROCs associated with it in order for it to be included as a watchstation requiring manning under one or more ship conditions. A special 4-digit code is used to specify the ROC element (see ROCTITLE), and the code '0000' indicates a watchstation which is always required under the ship conditions specified in the watchstation record, under any possible (real or hypothetical) ship ROC. It is important to note that the WSROC records may be classified CONFIDENTIAL in that they identify elements of a ship's ROC.

Data Source - The Ship ROC/Watchstation team gathered this data in the form of arrays showing each ROC element, and then identifying all watchstations requiring manning in order to satisfy the requirements of the ROC element. See Figure H-1.

This process involved assigning ROC ID numbers to the columns of a large matrix, Watchstation IDs to the rows, and then going down each column in turn to identify the watchstations required for the ROC element. Once the matrix is completed, the data is coded by WATCHSTATION, i.e., across the rows, rather than by column. Any watchstation (row) with more than 40 ROC elements driving it is coded as ROC element 0000, in that it will virtually always be a required watchstation.

Since WSROC transaction cards identify a ship and a group of ROC elements (albeit encoded), they may be classified CONFIDENTIAL. WSROC data is to be gathered, coded, and entered to the data base

only as directed by NAVMMACLANT upon resolution of the security question, by in-house or contractor personnel possessing the required security clearance, as directed by NAVMMACLANT.

Watchstations	ROC ELEMENTS					
	0000	0001	0002	0003	''''	0900
WS1	x					
WS2		x	x		x	x
WS3	x		x		x	
.						x
.		x	x	x		
.			x			
.					x	
.				x		
.	x					x
.		x			x	
WS300			x			x

Figure H-1. Watchstations Required by ROC Elements

I. WSTITLE

The Watchstation Title record identifies the nomenclature associated with a 10-digit Watchstation ID number.

Data Source - The original dictionary of approximately 6000 watchstation titles was compiled by the Ship ROC/Watchstation team from approved SMD II documents, and other sources, in consultation with NAVMMACLANT personnel. Updates and additions of watchstation titles and the assignment of WSID numbers is to be done in-house at NAVMMACLANT as needed.

Data Entity Inter-relationships

This paragraph defines the data entity relationships and, in addition, contains special notes relative to the processing of the system plus any additional notes about the relationships.

A. CLASS

1. The ship CLASS record is a header record which groups ships together. Every ship class has at least one ship, and each ship is a member of one and only one ship class. The set CLASS-SHIP contains all ships in the class.

2. The ship CLASS record is also a header for watchstations common to all ships of the class. For a watchstation record to belong to the CLASS-WS set, both the watchstation description (Watchstation ID number and title, and number of watchstanders) and the watchstander description (Rating, Rate, NEC, Org Code) must be identical for all ships of the class. Further, the same set of ROC elements must be related to the watchstation in terms of requiring its manning. The CLASS-WS set contains all such watchstations.

B. LEVEL

1. The LEVEL record serves as an identifier of a tasking level for a ship and the header for the ship's Required Operational Capabilities Statement (ROC) at that tasking level. The LEVEL-ROC set contains all ROC elements in the tasking level for the set.

2. The SHIP-LEVEL set contains all tasking levels for a given ship. The tasking level, in turn, contains the ship's Required Operational Capabilities Statement (ROC).

C. ROC

1. See B.1 for a description of the LEVEL-ROC set.

2. The ROC record is also a member of the ROCTITLE-ROC set, which links together tasking levels (and therefore ships and classes) which contain a given ROC element. This could be of use if a given ROC element were changed or deleted from the system, and it were necessary to determine the areas of impact. Keying on the SOCID for a ROC title gives the ROCTITLE header record. Then the ROCTITLE-ROC set can be stepped through to list the tasking levels and ships affected.

D. ROCTITLE

1. The ROCTITLE record identifies a ROC element and serves as header for the set of like ROC elements in all tasking levels. See C.2 for a description of the ROCTITLE-ROC set.

E. SHIP

1. See A.1 for a description of the CLASS-SHIP set.

2. See B.2 for a description of the SHIP-LEVEL set.

3. The ship record, in addition to heading all the tasking levels for a given ship (SHIP-LEVEL), also heads all watchstations for the ship which are not common to all ships of the class. This SHIP-WS set contains only those watchstations for the ship which are not already members of the CLASS-WS set. In order to see all watchstations assigned for a ship, it is necessary to pass through the entire SHIP-WS set for the given ship then the entire CLASS-WS set for the class containing the ship. The CLASS-SHIP set contains pointers to the CLASS record from each ship (owner pointers) to facilitate this process. See the CLASS-SHIP and CLASS-WS sets (A.1 and A.2, respectively) for further explanation.

F. WS

1. See A.1 for a description of the CLASS-WS set.

2. See E.3 for a description of the SHIP-WS set.

3. Each watchstation, in order to be manned at some ship condition, must be driven by one or more required operational or sub-operational capabilities (ROC elements) of the ship's Required Operational Capabilities Statement (ROC). The WS-WSROC set contains all the ROC elements which are capable of driving the watchstation, if they were part of the ship's ROC at the given tasking level. If any one of the ROC elements in the WS-WSROC set is also in the LEVEL-ROC set for the ship and tasking level of interest, the watchstation is required to be manned as specified in the WS record. Otherwise, the watchstation is omitted. See the LEVEL-ROC set for further details (B.1).

Pro-Forma Data Structures

This paragraph contains a description of the pro-forma data structure of the Ship ROC/Watchstation Module of NMRS, which is depicted in Figure H-2.

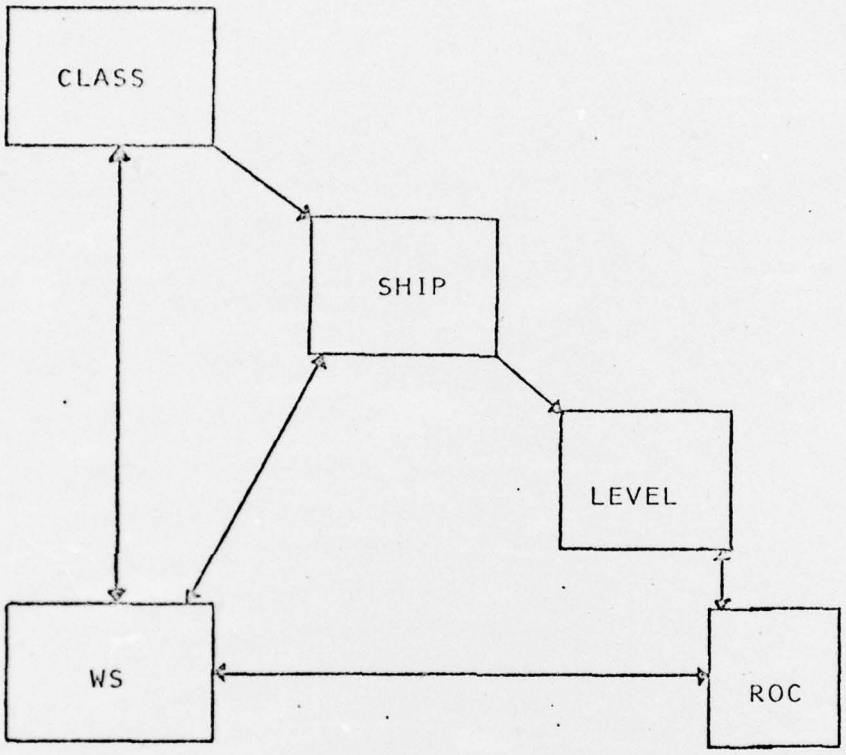


Figure H-2. Ship ROC/Watchstation Module Pro-Forma Data Structure.

A. CLASS-SHIP

Each ship class contains at least one ship, and each ship is a member of one and only one ship class.

B. CLASS-WS

Each ship class may contain one or more watchstations common to all ships of the class. A watchstation may or may not belong to a class; it may, instead, be peculiar to one or more ships of the class, but not all of them. In that case, it belongs to the SHIP-WS set, and not to the CLASS-WS set. Where a watchstation is the same within several classes and/or ships, it is separately stated within each.

C. SHIP-WS

Each ship may contain one or more watchstations which are not common to all ships of the class. Where a given watchstation occurs for several, but not all, ships of the class, the watchstation record is repeated as necessary for each applicable ship. A watchstation may or may not belong to a ship; it may, instead, be common to all ships of the class. In that case, it belongs to the CLASS-WS set, and not to the SHIP-WS set.

D. SHIP-LEVEL

Each ship may possess one or more tasking levels, which are identified by ship and level ID. Each tasking level belongs to one and only one ship. Different ships may have identical tasking levels, but their identification by ship will differ, and all information will be separately stated.

E. LEVEL-ROC

Each tasking level may contain one or more ROC elements, and each ROC element may appear in one or more tasking levels. For simplicity of processing, each tasking level has a separately stated list of ROC elements. Because the ROC record is at the minimum size for IDMS records, no disk storage savings can be effected through the use of link records.

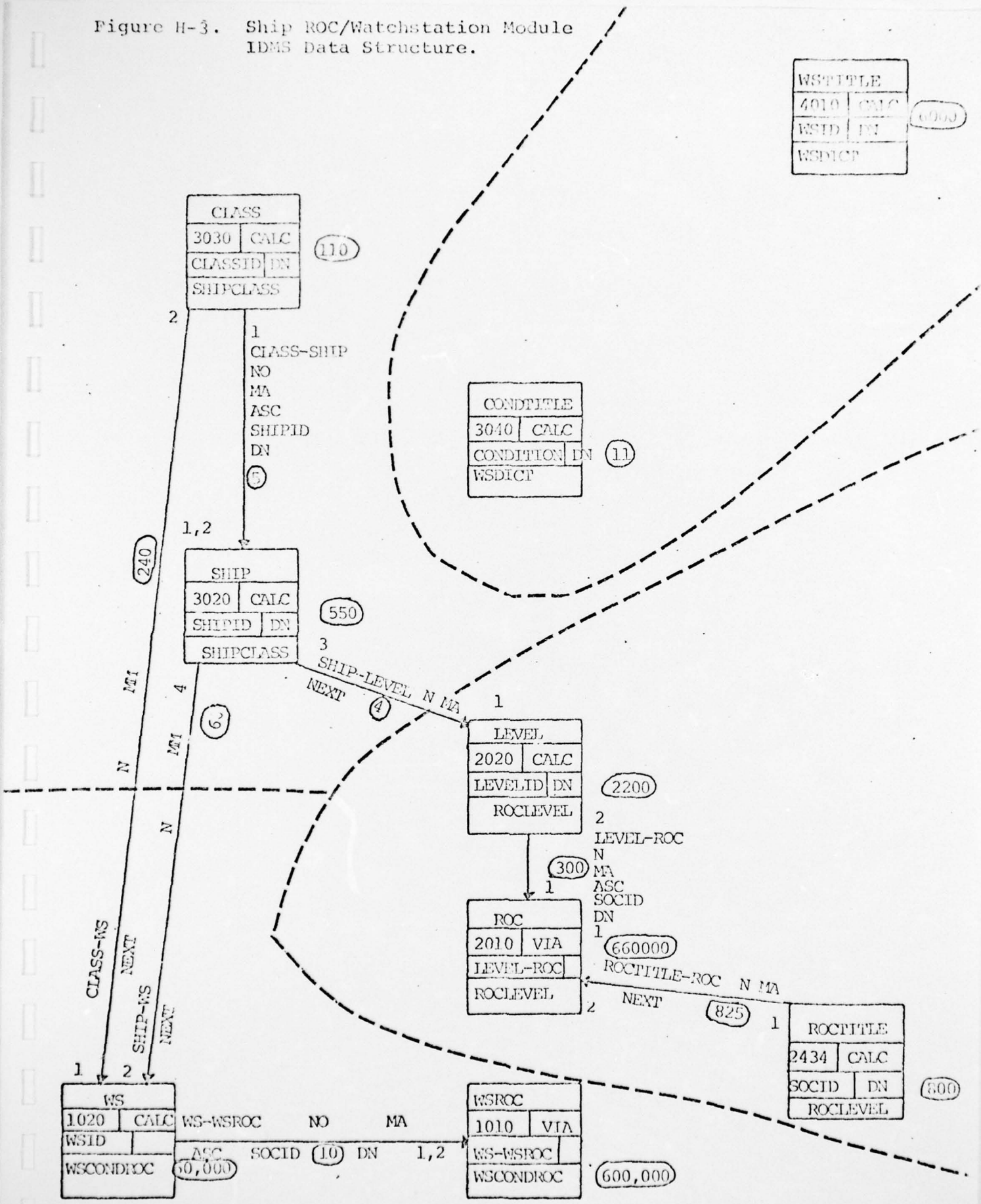
F. WS-ROC

Each watchstation may be driven by one or more ROC elements, and each ROC element may drive one or more watchstations. Because the WSROC record is at the minimum size for IDMS records, no disk storage savings can be effected through the use of link records. Therefore, each driving ROC is separately stated for each watchstation.

IDMS Data Structure

This paragraph briefly describes the IDMS data structure of the Ship ROC/Watchstation Module of NMRS, as depicted in Figure H-3.

Figure H-3. Ship ROC/Watchstation Module IDMS Data Structure.



DATA BASE DESIGN

Schemas

Currently, only one schema exists for the Ship ROC/Watchstation Module.

A. SHPROCWS

Other test versions of the schema are expected when the Ship ROC/ Watchstation Module database is incorporated within the NMRS database.

DMCLs

Currently, only one Device Media Control Language (DMCL) exists for the Ship ROC/Watchstation Module:

A. DMUPDATE

As the purpose of the DMCL is to fine-tune system performance in production mode, additional test versions of the DMCL are expected when the Ship ROC/Watchstation Module goes into production on a secure computer facility.

Subschemas

Currently, only one Subschema exists for the Ship ROC/Watchstation Module:

A. BUGTESTR

As this is sufficient for all processing currently being done as well as all identified modifications, no additional subschema are envisioned.

Database Files

Two database files currently exist for the Ship ROC/Watchstation Module:

A. SROCWSMN

B. SROCWSRC

This data structure is divided into four realms:

(1) WSDICT,
(2) SHIPCLASS,
(3) WSCONDROC, and
(4) ROCLEVEL

The WSDICT realm contains dictionary-type information or watchstation titles and ship condition titles. The ship class realm contains ship and class dictionary information, and their interrelationships. The WSCONDROC realm contains watchstations by ship conditions, and the ROC elements which drive them. The ROCLEVEL realm contains tasking levels and the ROC elements comprising them, as well as a dictionary of ROC element titles.

As the database files are the physical basis for the logical database, they are highly dependent on the nature of the host computer facility and its restrictions. Changes to the sizes of these files are certain, as they are merely small test data sets at present. The quantity of files, and their names, may well change upon integration into the NMRS database, or upon transfer to another computer facility. These changes are completely transparent to the application programmer and system user; they are solely the responsibility of the Database Administrator and his staff.

Realms

Four Realms currently exist in the Ship ROC/Watchstation Module:

A. WSDICT

B. SHIPCLASS

C. WSCONDROC

D. ROCLEVEL

These realms serve to group data functionally, aid in system testing, and to enhance system performance. No changes to Realms are envisioned, although such changes would be transparent to the existing application programs and the system users.

Record Types

Nine record types currently exist in the Ship ROC/Watchstation Module:

A. CLASS

B. CONDTITLE

C. LEVEL

D. ROC

E. ROCTITLE

F. SHIP

G. WS

H. WSROC

I. WSTITLE

Sets

Seven sets currently exist in the Ship ROC/Watchstation Module:

A. CLASS-SHIP

B. CLASS-WS

C. LEVEL-ROC

D. ROCTITLE-ROC

E. SHIP-LEVEL

F. SHIP-WS

G. WS-WSROC

Miscellaneous.

a. Special programs to list any record type or set in the database. (Only if commands/reports to do so are not part of the deliverables).

A number of IDMS/CULPRIT report programs are included:

A. Condition Title Report
B. ROC Title Report
C. ROC Report
D. Watchstation Titles Report
E. Watchstation/ROC Relationship Report

These reports are further documented in Culprit Reports of the Ship ROC/Watchstation Database.

b. Location (Dataset Name and Volume) of all files (source code, object code, etc.).

All files are given names with the prefix WEUIDAX.MSS.SHIPROC to clearly identify the group responsible for their maintenance. Following is a list of all datasets, with their volume serial numbers, currently maintained by the Ship ROC/Watchstation Team at NIH (prefix omitted here):

A.	CLUCARDS	FILE16
B.	DATADICT	FILE16
C.	DATALIB	PDS004
D.	IDMSLIB	FILE16
E.	LOADLIB	PDS004
F.	ROC	FILE16
G.	SRCELIB.COBOL	PDS004
H.	SRCELIB.FORTRAN	PDS004
I.	UTCNTROL	FILE16
J.	WS	FILE16
K.	WSMAIN	FILE16
L.	N.DATADICT	FILE16
M.	N.IDMSLIB	FILE16
N.	N.LOADLIB	PDS004
O.	N.ROC	FILE17
P.	N.SRCELIB.COBOL	PDS004
Q.	N.WSMAIN	FILE17

DISTRIBUTION LIST

One copy to each addressee except as otherwise noted:

Deputy and Chief Scientist (Code 102)
Office of Naval Research
Arlington, VA 22217

Assistant Chief for Technology (Code 200)
Office of Naval Research
Arlington, VA 22217

Director of Technology (Code 201)
Office of Naval Research
Arlington, VA 22217

Assistant Chief for Research (Code 400)
Office of Naval Research
Arlington, VA 22217

Director of Research (Code 401)
Office of Naval Research
Arlington, VA 22217

Manager, Program in Manpower R&D (Code 450) - 12 copies
Office of Naval Research
Arlington, VA 22217

Head, Operations Research Branch (Code 434)
Office of Naval Research
Arlington, VA 22217

Contract Administration Branch (Code 622)
Office of Naval Research
Arlington, VA 22217

Naval Research Laboratory
Technical Information Division
4555 Overlook Avenue, S.W.
Washington, D. C. 20375

Research Pyschologist
Office of Naval Research Branch Office
536 South Clark Street
Chicago, IL 60605

Psychologist
Office of Naval Research Branch Office
495 Summer Street
Boston, MA 02210

Psychologist
Office of Naval Research Branch Office
1030 East Green Street
Pasadena, CA 91106

Defense Documentation Center - 12 copies
Attention TC
Building 5
Cameron Station
Alexandria, VA 22314

Head, Manpower Training and Reserve Group (Op-964D)
Room 4A538, The Pentagon
Washington, D. C. 20350

Manpower Analysis and Systems Development Branch (Op-121)
Room 1606, Arlington Annex
Washington, D. C. 20370

Human Resources Program Manager
Naval Material Command (0344)
Room 1044, Crystal Plaza #5
2221 Jefferson Davis Highway
Arlington, VA 20360

Technical Director - 5 copies
Navy Personnel Research and Development Center
San Diego, CA 92152

Scientific Advisor to the Chief of Naval Personnel (Pers Or)
Naval Bureau of Personnel
Room 1416, Arlington Annex
Washington, D. C. 20370

Special Assistant for Enlisted Force Analysis
Naval Bureau of Personnel (Pers-2x)
Room 2628, Arlington Annex
Washington, D. C. 20370

Assistant Deputy Chief of Naval Personnel for
Retention Analysis and Coordination (Pers-12)
Room 2403, Arlington Annex
Washington, D. C. 20370

Military Assistant for Human Resources
Office of the Director of Defense Research & Engineering
Room 3D129, The Pentagon
Washington, D. C. 20301

Personnel Analysis Division
AF/DPXA, Headquarters USAF
Room 5C360, The Pentagon
Washington, D. C. 20330

Technical Director
U. S. Army Research Institute for the
Behavioral and Social Sciences
1300 Wilson Boulevard
Arlington, VA 22209

Program Director
Manpower Research and Advisory Services
Smithsonian Institution
801 N. Pitt Street
Alexandria, VA 22314

Director, Management Information Systems Office
OSD(M&RA)
3B917, The Pentagon
Washington, D. C. 20301

Dr. Lorand B. Szalay
American Institutes for Research
3301 New Mexico Avenue, N. W.
Washington, D. C. 20016

Mr. Richard K. Hovey
B-K Dynamics
15825 Shady Grove Road
Rockville, MD 20850

Dr. James P. Murphy
Booz-Allen Applied Research
4733 Bethesda Avenue
Bethesda, MD 20014

Dr. Richard S. Hatch
Decision Systems Associates, Inc.
5640 Nicholson Lane
Rockville, MD 20852

Mr. Clinton W. Kelley, III
Decisions and Designs, Inc.
8400 Westpark Drive, Suite 600
McLean, VA 22101

Mr. Daniel F. Huck
General Research Corporation
Operations Analysis Division
7655 Old Springhouse Road
McLean, VA 22101

Dr. Robert Vineberg
HumRRO Western Division
27857 Berwick Drive
Carmel, CA 93921

Dr. Joseph Augusta
Ketron, Inc.
1400 Wilson Boulevard
Arlington, VA 22209

Commanding Officer
Navy Manpower and Material Analysis Center, Atlantic
U. S. Naval Station
Norfolk, VA 23511

Commanding Officer
Navy Manpower and Material Analysis Center, Pacific
San Diego, CA 92132